ISKOLASZÁMÍTÓGÉP

Kovács Imre
Szabó Gizella–Tick József
Tiszai Tamás

ABC 80
kezdőknek

TUDOMÁNYSZERVEZÉSI
ÉS INFORMATIKAI INTÉZET
Budapest 1982

ISKOLASZÁMITÓGÉP SOROZAT

1987 -11- 1 3

KOVÁCS IMRE – SZABÓ GIZELLA – TICK JÓZSEF – TISZAI TAMÁS:

ABC 80 KEZDŐKNEK

TUDOMÁNYSZERVEZESI ES INFORMATIKAI INTÉZET
Budapest, 1982.

Az ISKOLASZÁMITÓGÉP sorozat első kötete: ABC 80 kezdőknek

Készítették a Kandó Kálmán Villamosipari Műszaki Főiskola
Matematikai és Számítástechnikai Intézetének munkatársai.

Sorozatszerkesztő: Zsáry Piroska

1996
2007 MÁJ
Ipari Szakmunkás
KÖNYVTÁR
4704/1
Intéz

ISBN 963 451 114 1

Kiadja a Tudományszervezési és Informatikai Intézet
Felelős kiadó: Páris György igazgató
Készült az Országos Pedagógiai Könyvtár és Múzeum nyomdájában, 2000 példányban.
Munkaszám: 82/133
Felelős vezető: Erdélyi István

E l ő s z ó

A Számitástechnikai Tárcaközi Bizottság által 1979. julius 29-én elfogadott ajánlás értelmében a középfoku oktatási intézményekben általánossá kell tenni a számitástechnika és a számitástechnika alkalmazásának oktatását. Cél, hogy 1990-ben már ne végezzen olyan középiskolai tanuló, aki nem kapott számitástechnikai oktatást. "Iskolaszámitógép" cimü sorozatunkkal az oktatás céljára alkalmas mikroszámitógépek alkalmazásának elsajátitását szeretnénk elősegiteni.
A sorozat első kötete a már használatban lévő ABC-80 számitógépek alkalmazásához kiván segitséget nyujtani.

Budapest, 1982. junius 29.

TARTALOMJEGYZÉK

BEVEZETÉS 5
1. Az ABC-80 KEZELÉSE 6
1.1. Az ABC-80 felépitése és müködtetése 6
1.1.1. Az ABC-80 üzembehelyezése 6
1.1.2. A klaviatúra használata 7
1.1.3. A monitor 10
1.1.4. Kazettás tároló 10

1.2. Konstansok 11
1.2.1. Numerikus konstansok 12
1.2.1.1. Egész konstansok 12
1.2.1.2. Valós konstansok 13
1.2.2. Szövegkonstansok 15

1.3.Változók 16
1.3.1. Aritmetikai változók 16
1.3.1.1. Egész változók 16
1.3.1.2. Valós változók 17
1.3.2. Szöveges változók 17

1.4. Kifejezések 17
1.4.1. Aritmetikai kifejezések 18
1.4.1.1. Egész kifejezések 21
1.4.1.2. Valós kifejezések 21
1.4.2. Szöveges kifejezések 22

1.5. Standard függvények 22

1.6. Direkt üzemmód 23

1.7. Programkezelés 25
1.7.1. Program betöltése kazettáról 25
1.7.2. Program elinditása, futtatása 27
1.7.3. Program felvétele kazettára 27

2. Az ABC-80 BASIC alaputasitásai 29
2.1. BASIC programok irásának formai szabályai 29
2.2. Értékadás - a LET utasitás 31
2.3. Adatbevitel - az INPUT utasitás 31
2.4. Nyomtatás - a PRINT utasitás 33
2.5. Programzárás - STOP, END utasitás 36
2.6. Feltételes vezérlésátadás - IF-THEN-ELSE utasitás 37
2.7. Feltétel nélküli vezérlésátadás - GOTO utasitás 42
2.8. Megjegyzések elhelyezése - REM utasitás 44
2.9. Ciklusszervezés - FOR...NEXT utasitás 44
2.1o. Adatblokk használata - READ, DATA, RESTORE utasitások 53
2.11. Szubrutinszervezés - GOSUB...RETURN utasitások 57
2.12. Függvény-definiálás - DEF FN utasitás 59
2.13. Tömbök - DIM utasitás 62
2.14. Programfutás megállitása - a GET utasitás 66

3. Az ABC-80 rendszer parancsai 67
3.1. Programok inditása - RUN parancs 67

3.2. Törlés 68
3.2.1. CLEAR parancs 68
3.2.2. NEW parancs 68

3.3. Programok kiiratása - LIST parancs.....68

3.4. Programsorok átszámozása - REN parancs 71

3.5. Hibakeresés és javitás 73
3.5.1. Az ED parancs 73
3.5.2. A TRACE parancs 75
3.5.3. A NOTRACE parancs 75

3.6. Programok mentése és betöltése 76
3.6.1. A SAVE parancs 76
3.6.2. A LOAD parancs 77

4. Speciális programozási lehetőségek 79
4.1. Grafikus üzemmód használata 79
4.1.1. Átváltás grafikus üzemmódba 79
4.1.2. Rajzolás a képernyőre grafikus utasitá-
sokkal 79
4.1.3. Rajzolás a képernyőre fél-grafikus ka-
rakterekkel 80

4.2. Hangeffektusok előállitása 83

4.3. A V24-soros interface programozása 84

4.4. A kettős kazettás adattároló kezelése 86

4.5. A gépi szintű programozást támogató utasitások..87

4.6. ASCII aritmetika 89

4.7. Logikai kifejezések 91

5. File kezelés 93
5.1. File nyitás és zárás 93

5.2. Irás illetve olvasás a file-ból 96

5.3. Programok rögzitésére és betöltésére szolgáló
parancsok 98

5.4. Speciálisan diszk egységre értelmezett parancsok. 10

6. Néhány programszerkesztési recept 103
6.1. Adatbevitel 103
6.1.1. Egyedi adatbevitel 103
6.1.2. Összetartozó adatok bevitele 104

6.2. Kiirás 107
6.2.1. Képernyő törlés 107
6.2.2. Cursor pozicionálás 107
6.2.3. Hosszú listák kiirása 108

6.3. Megjegyzések 109

Függelékek 110
F1. Rendszerparancsok, rendszerutasitások 111
F2. Rendszerfüggvények 123
F3. Hibajelzések 126

B E V E Z E T É S

Egy mikroszámitógép kezelési kézikönyv ritkán tud minden igényt kielégiteni, hiszen az igények igen szélsőségesek lehetnek. Vannak olyanok, akik egy kézikönyvből szeretnék elsajátitani a számitástechnikai kultura alapelemeit is, mások viszont csak a specifikumok, az uj, ill. a meglévő, megszokottól eltérő dolgok iránt érdeklődnek. Az ABC-80 gép ismertetése, programozási lehetőségeinek bemutatása során igyekszünk a gépet felhasználók minél szélesebb érdeklődési spektrumát lefedni, igy részletesebben, (de modellszerüen) tárgyaljuk az ABC-80 kezelése fejezetben az adatábrázolást és alapelemeket, a második fejezetben pedig néhány példával illusztrálva az ABC-80 magasszintü programozási nyelvének, a BASIC-nek (ejtsd: bézik) általános utasitásait. A harmadik fejezet az ABC-80 parancsaival, mig a negyedik a speciális programozási lehetőségekkel foglalkozik. Az ötödik fejezetben leirjuk az ABC-80 file kezelését, a hatodikban pedig a kezdő programozóknak adunk néhány receptet programjaik szerkesztéséhez.

A gépet megismerőknek és már kezelni ill. programozni tudóknak a Függelék lesz az igazán hasznos, mert ott összefoglalóan is megtalálják az ABC-80 BASIC utasitásait, függvényeit valamint rendszerparancsait. Ugyancsak a függelékben található a programkészités és kipróbálás, futtatás során talált hibákra vonatkozó hibakódok jegyzéke.

Reméljük, hogy ismertetőnk minden jövőbeli olvasónak hasznos elméleti és gyakorlati tanácsokkal, utbaigazitásokkal szolgál.

1. Az ABC 80 kezelése

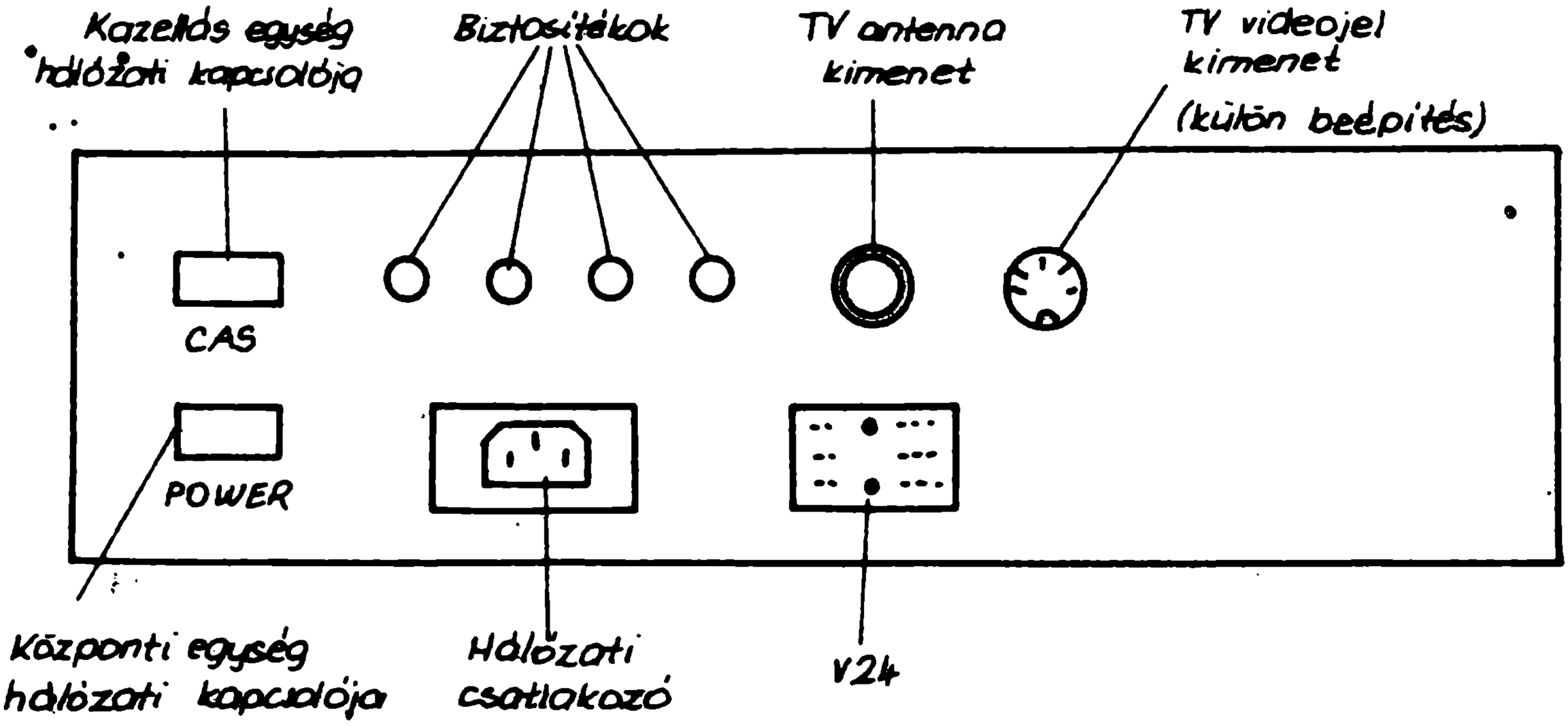
1.1 Az ABC 80 felépitése és működtetése

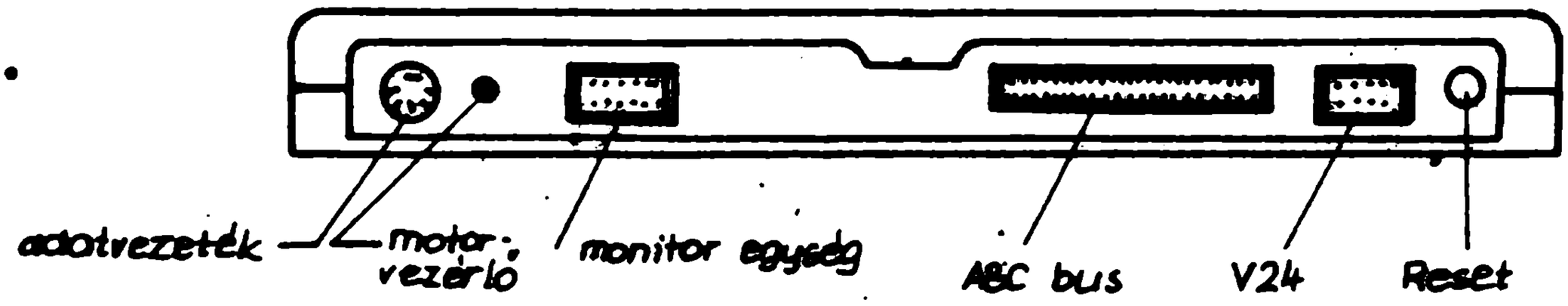
Az ABC 80 mikroszámitógép három fő részből áll:

- klaviatúra (billentyűzet) és központi egység.
- monitor (képernyő) - valamilyen kommersz TV vevőkészülék,
- kazettás (egy vagy két egységes) tároló egység.

1.1.1 Az ABC 80 üzembehelyezése.

Csatlakoztassuk a kazettás egységet a központi egységhez. Az összekapcsoláshoz az 1.1 ábrán bemutatjuk a csatlakozóhelyeket.





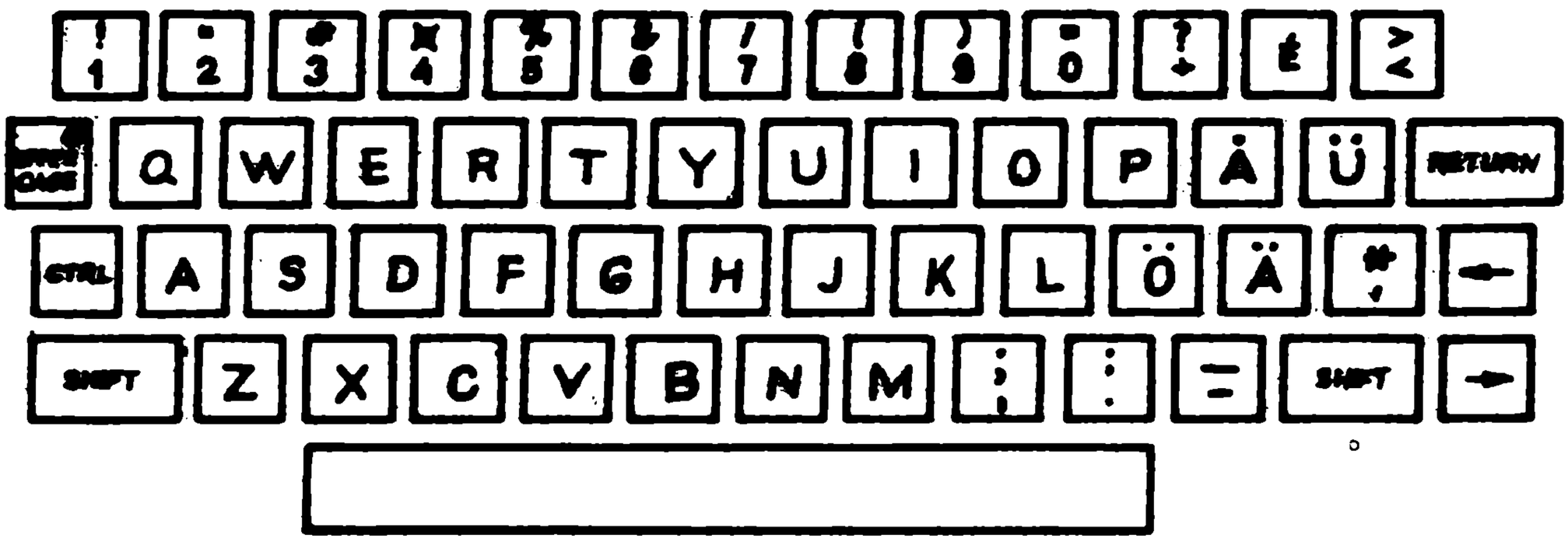
1.1. ábra. Kettős kazettás tároló és klaviatura hátlapja a csatlakozóhelyekkel.

A monitor egység antenna csatlakozó kábelét csatlakoztassuk a kazettás egység hátoldalán található antenna csatlakozó aljazatba (TV feliratú csatlakozóhely)
A kazettás adattároló egység 5 pólusú DIN szabványú csatlakozóját dugjuk be a klaviatúra hátoldalán található aljazatba, ezt követően pedig a magnomotor vezérlőkábelét csatlakoztassuk.
A monitort és a kazettás adattároló egységet csatlakoztassuk a 220 V feszültségű hálózathoz.

- Monitor: a TV készülék szokásos kezelőszerveit kell használni bekapcsolásra, fényerősség szabályozására. Éles kép beállitása a készüléknek az UHF sáv 40-es csatornára hangolásával történik.
- Kazettás egység: Bekapcsolása az egység hátlapján lévő CAS nyomókapcsolóval lehetséges. Ezt követően az előlapon található CAS1 kijelző világitani fog.

1.1.2. A klaviatúra használata

A klaviatúrán fehér és narancssárga szinű billentyűket találunk. A fehér szinüek szolgálnak a használható karakterek (jelek) gépbe való bevitelére (és a monitoron való megjelenitésre), mig a narancssárgák speciális funkciókat látnak el. A klaviatúrát a 1.2. ábrán mutatjuk be.



1.ℓ. ábra.

A használható karakterek billentyűi.

- **az angol ABC 26 kis- és nagy-betűje:** a, A, b, B, c, C, d, D, e, E, f, F, g, G, h, H, i, I, j, J, k, K, l, L, m, M, n, N, o, O, p, P, q, Q, r, R, s, S, t, T, u, U, v, V, w, W, x, X, y, Y, z, Z
- **5 ékezetes kis- és nagybetű:** ä, Ä, å, Å, ö, Ö, ü, Ü, é, É
- **1ø számjegy:** ø, 1, 2, 3, 4, 5, 6, 7, 8, 9
- **speciális karakterek:** ! " # ¤ % & / ( ) = ? + < > * ' - ; , _ **(szóköz karakter)**

Ismertetőnkben - mint ahogy az a számitástechnikában szokásos - a nullát az O betűtől való megkülönböztetés érdekében áthuzással ø jelöljük.

Speciális billentyűk

SHIFT

Lenyomott állapotában a vele együtt lenyomott fehér billentyű felső részén található karaktert állitja elő. A betűket tartalmazó billentyűkön csak egy jel van, ezeket a SHIFT-tel együtt nyomva nagybetűk állithatók elő. A SHIFT billentyűk az alsó billentyűsor mindkét oldalán megtalálhatók és egymástól függetlenül használhatók.

UPPER CASE

Ez a billentyű öntartó, azaz, ha benyomjuk, lenyomva marad és ezt az állapotát a billentyűbe épitett piros jelzőfény mutatja. Hatást csak a 26 + 5 betűt jelentő billentyűre gyakorol - lenyomott állapotában ugyanis mindig csak nagybetűk irhatók be ezekkel a billentyűkkel. Ha vissza akarunk váltani normál (tehát SHIFT nélkül kisbetűket iró) állapotba, akkor a billentyűt ismét meg kell nyomni.

CTRL

A CTRL billentyű a vele egyidőben lenyomott billentyű karakterkódját úgy módositja, hogy a gép azt közvetlen vezérlőjelnek értelmezze. Az értelmezett vezérlőjelek:

CTRL C : futó program vagy listázás megszakitása
CTRL X : a cursor a sor elejére ugrik vissza,
CTRL ξ : a cursor helyén egy besatirozott négyzet keletkezik

⇐

Lenyomáskor a cursor egy hellyel balra lép. Folyamatosan lenyomva tartva 10 Hz-el ismétli a léptetést.

⇒

Lenyomáskor a cursor egy hellyel jobbra lép. Folyamatosan is használható.

RETURN

A sor lezárására szolgáló billentyű. Folyamatosan is használható. Folyamatos lenyomásakor a monitoron látható kép felfelé tolódik el, mig végül csak a villogó cursor marad meg.

1.1.3. A monitor

Az ABC 80 monitoraként bármilyen képernyős egységet is használjunk, az 24 sorban soronként 40 karakter megjelenitésére ad lehetőséget. A kiirt jelek nagysága a képernyő méretétől függ. Az egyes karakterek egy 9 x 7-es pontmátrixban a karakter alakjának megfelelő világos pontok összességéből állnak elő.
Az ABC 80 grafikus üzemmódban is használható, ilyenkor a képernyő területének felosztása finomabb- összesen 72 sor és 79 oszlop, azaz összesen 5760 képelem képezhető. A grafikus üzemmódot részletesebben a 4.1. pontban ismertetjük.

Az ABC 80 hangeffektusok keltésére is képes, ezeket a monitor hangszórójának közvetitésével észlelhetjük. Az ABC 80 hangkeltési lehetőségeinek leirását a 4.2. pontban oldjuk meg.

1.1.4. Kazettás tároló

Az ABC 80 gépekhez egy vagy két kazettás tároló egységet használhatunk. Az egy kazettás egység bekapcsolás után azonnal üzemképes, mig a két kazettás változatnál az 1. számú egység lesz automatikusan

működőképes állapotban. Két kazettás egységnél egyidőben csak az egyik (vagy az 1. vagy a 2.) használható.
A kazettás tárolók kezelőszervei és azok funkcióira a számrakosztató elektronikában használt kazettás magnókéval megegyeznek, csakhogy itt a kazettákra adatokat és programokat vesszünk fel, ill. játszunk le.

Adat ill. programrögzítésre legcélszerűbb a többszörösen ellenőrzött Computer Casette-ák használata, de jó minőségű kommersz kazettákat is igénybe vehetünk ilyen célokra. Fontos azonban az, hogy Cr_2O_3 kazettát ne használjunk, mert azok a nem különösen kemény anyagú felvevő/lejátszó fejek gyors kopásához vezetnek.
A kazettás egységekről történő programbevitelt ill. a kazettára történő programkimentést a 3.6 pontban irjuk le részletesebben.

1.2. Konstansok

Akármilyen kis gép is az ABC 80, felhasználásának lényegét tekintve megegyezik a nagy számitógépek által adott lehetőséggel: adatokat dolgoz fel, azaz adatok transzformációját végzi.
A feldolgozásra szánt adatokat a gépben tárolni kell. A tároló nagyságának, adatbefogadó képességének jellemzésére a tárkapacitás fogalmát használjuk. Ennek méréséhez azt adjuk meg, hogy a tárolóban összesen hány db kettes számrendszerbeli számjegyet (Ø-t, vagy 1-et, azaz összefoglaló néven bitet) helyezhetnénk el, ugyanis a gép minden tárolt információt kettes számrendszerbeli számként ábrázol. Hogy hogyan is lesz minden adatunkból kettes számrendszerbeli szám és ezeket pontosan hogyan tartja nyilván az ABC 80, az

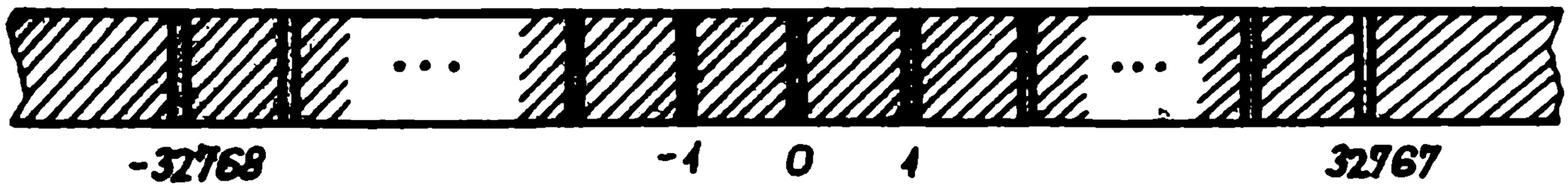
számunkra, akik csak többnyire számolási feladataink megoldására akarjuk használni a gépet, nem túl érdekes. Azt azonban érdemes megjegyeznünk, hogy a biteket ábrázoló tárolócellákat nagyobb csoportokba fogják össze a könnyebb kezelhetőség érdekében, és igy kapják a 8 bitet magába foglaló byte (bájt) elnevezésű tárolóegységet. Az ABC 80 tárolójának kapacitása 16KB = 16 Kilobyte = 16.2^{10}byte (mert a számitástechnikában 1 Kilo = 2^{10} = 1024-et jelent)= = 16 . 2^{10} . 8 bit. Az adattárolásról ennyit feltétlenül tudnunk kell, mert csak igy érthetjük meg a különböző adattipusok felhasználási lehetőségeit.
A feladatmegoldások során olyan adatokkal dolgozhatunk, amelyekkel számolási műveleteket kell végrehajtanunk - ezek a számok vagy numerikus adatok -, ill. olyanokkal, amelyekkel nem kell számolni, csak tájékoztató, értelmezést megkönnyitő szövegként használatosak.
A következőkben ezeket az adattipusokat részletesebben tárgyaljuk.

1.2.1. Numerikus konstansok

Az ABC 80 számára a számértékkel rendelkező, számjegyekkel megadott és a feldolgozás során változatlan adatokat numerikus konstansoknak nevezzük. A numerikus konstansokat az ábrázolásukhoz felhasznált tárolóterület hossza szerint két csoportba oszthatjuk.

1.2.1.1. Egész konstansok

Az egész tipusú konstans tárolásához biztositott tárterület hossza 2 byte, ez viszonylag kicsi, ezért az egész konstansok értelmezése elég kis intervallumra korlátozódik: (ld.13. ábra)



1.3. ábra.

Az egész konstansok a -32768, 32767 intervallum egész értékű pontjai, igy természetes, hogy felirásukhoz csak az előjel (+vagy -) és a számjegyek szükségesek, Annak jelzésére azonban, hogy ezeket az adatokat a gépnek rövid tárolóterületen (2byte) kell elhelyeznie, az egyes helyértékű (jobboldali utolsó) számjegy után egy % jelet kell elhelyezni.
A számok megadásakor a + előjelet nem kötelező kiirni.

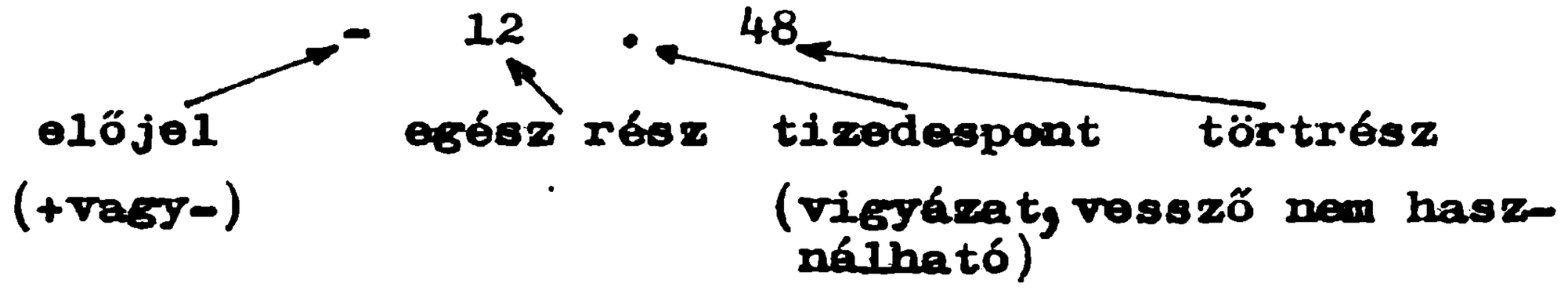
Helyesen felirt egész konstansok: 83% -12425%, 10%

1.2.1.2. Valós konstansok

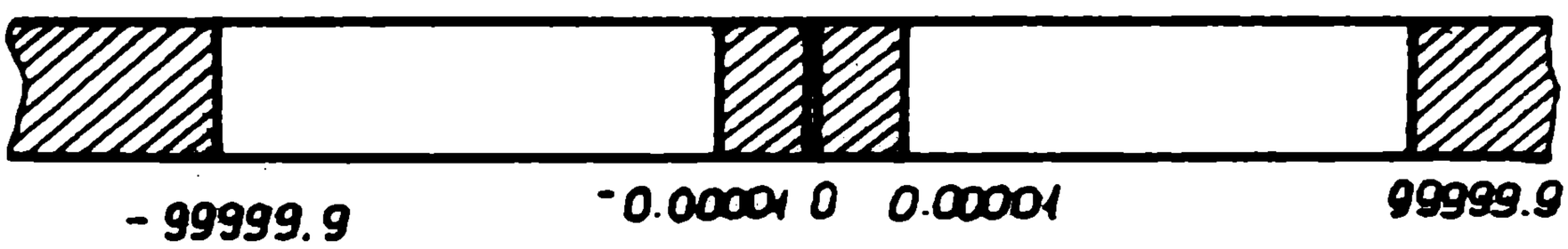
A valós konstansok tárolása számára hosszabb tárolóterületet használhatunk (5 byte-ot) ezért ezzel a konstanstipussal nagyobb számtartományt tudunk ábrázolni. A valós konstansoknak kétféle megadási ill. kiirási lehetősége van:

- szokásos számirási változat,
- hatványkitevős (un. féllogaritmikus) változat.

A szokásos számirási változatban a teljes szám a következő részekből áll:



A szám komponensei közül a pozitiv előjelet nem szükséges kiirni, az egész rész és a törtrész közül is csak azt, amelyik a számban előfordul. Ha törtrész nincs, akkor a tizedespont is elmaradhat.
Az ilyen formában felirt számokat az ABC 80 hat számjegy pontossággal tudja ábrázolni, igy a használható számábrázolási intervallum:(1.4.ábra)



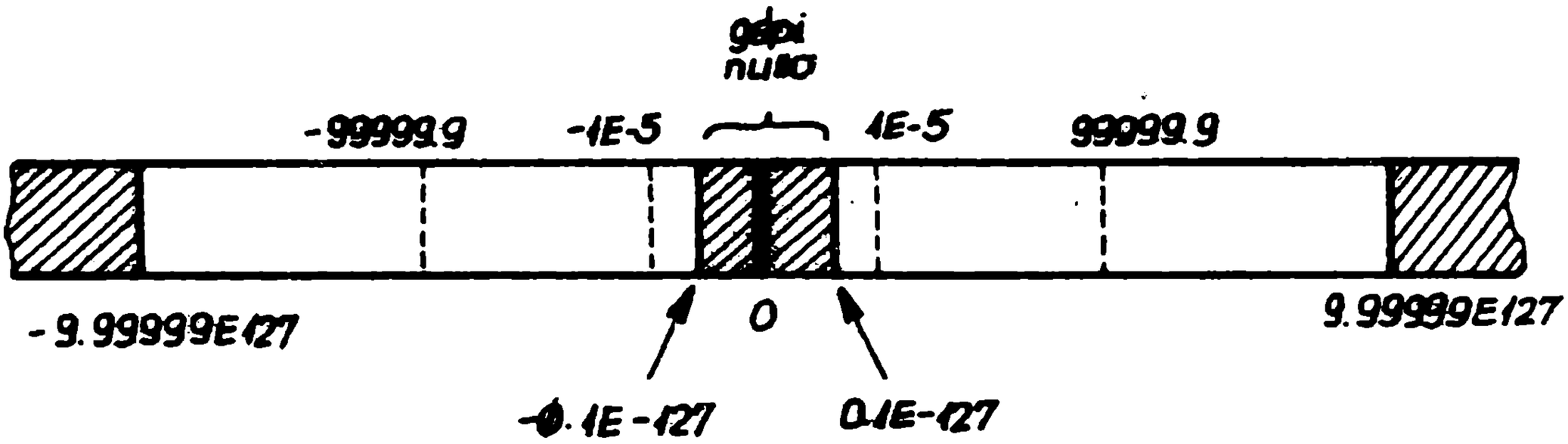
1.4. ábra.

Helyesen felirt szokásos alakú valós konstansok:
+23,1, -5, 4352.78, -22,53. Hatványkitevős számirás esetén az előző változathoz képest az az eltérés,hogy a számkomponensek között kitevőrész is szerepel:

$$-\ 1248\ \underbrace{E\ -\ 2}_{\text{kitevőrész}}$$

A kitevőrész azt mutatja meg, hogy a szokásos alakú számot 10 hányadik hatványával kell megszorozni ahhoz, hogy a szám valódi értékét megkapjuk. A szokásos szám és a kitevőt jelentő szám közé be kell irni az E karaktert. A hatványkitevős alakú szám komponensei közül (a szám előjele, egész része, tizedespont,törtrész, e karakter,kitevő előjele, egész értékű kitevő) a pozitiv előjelek elhagyhatók, az egész rész és a törtrész közül elég csak az egyiknek szerepelni, tizedespont csak akkor kell, ha törtrész is van.

A hatványkitevős ábrázolás a következő intervallumra érvényes: (1.5. ábra)



1.5. ábra.

Az 5. ábrán megjelöltük a szokásos számábrázolási formával leirható számtartományokat is.
A valós konstans begépelésekor a két valós számábrázolási mód mindegyikét használhatjuk az intervallum megkötöttségekre való tekintet nélkül. A gép az adatok tárolását mindig hatványkitevősen, a kiirásokat azonban az előzőkben bemutatott intervallumhatároknak megfelelően végzi, igy ha pl. 135ØØØØ-et gépelünk be, ennek a számnak a későbbi visszairása 1.35E+7 lesz, mig az 1.2E1 begépelése után, a tárolt szám kiirásakor 12-t kapunk.
Helyesen felirt hatványkitevős alakú valós konstansok: 1E5 (= 1ØØØØØ),.23E3 (=23Ø), 25E-2(=Ø . 25)

1.2.2. Szövegkonstansok

Szövegkonstansnak az idézőjelek közé tett betűket, számokat és speciális karaktereket tartalmazó sorozatot nevezzük. Az ABC 8Ø a szövegkonstansok határolására az idézőjeleken kivül az aposztrofot (')is megengedi. A szövegkonstansok egyenkénti tárolására biztositott tárterület legfeljebb 8Ø
jelet tartalmazó konstansok képzését teszi lehetővé, a pontos tárterület méretet a szöveg hossza adja; ahány karaktert a szöveg tartalmaz, annyi byte lesz a tárolási hossz.

Helyesen felirt szövetkonstansok: "ABC 8ø ",'LEGJOBB'

1.3. Változók

A változó olyan karaktersorozattal (változónévvel) azonositott tárolóterület, amelynek a mérete megfelel a használt adattipusnak és tartalma (értéke) a feladat megoldása során változhat.
Gondoljunk a 2r matematikai kifejezésre. Erről a kifejezésről tudjuk azt, hogy az r sugarú kör kerületének mérőszámát adja meg, bármilyen nem negativ szám is r értéke.

Az r tehát mindig a kör sugarát jelenti attól függetlenül, hogy konkrét értéke mennyi is lesz egy számolás elvégzésekor, azaz az r névvel azonositottunk egy adatot. A számitógépben az adatot valahol a tárolóban (memóriában) helyezzük majd el, az adatra hivatkozni pedig helyének megadásával tudunk. Az adat helyének megadására szolgáló terület neve a változó. Ugyanezen a tárolóterületen különböző időben különböző adatokat tárolhatunk, de a helyhez rendelt név - a változó-közben állandó is maradhat.
A változó értéke ezek szerint minden időpillanatban valamilyen konstand, de állandósága nem ennek a konstansnak van, hanem a konstans jelentésének (tárolóhelyének). A változók ezek szerint éppen olyan tipusokba sorolhatók, mint a konstansok.

1.3.1. Aritmetikai változók

Az aritmetikai változók olyan tárolóhelyeket azonositanak, melyekben numerikus konstansok helyezhetők el.

1.3.1.1. Egész változók

Az egész változók jelölésére használható változónevek (azonositók) legalább két karaktert kell hogy tartalmazzanak. Ilyenkor

- az első jel csak betű lehet,
- a második jel a % jel.

Az egész változót jelző név legfeljebb 3 karakteres lehet, ilyenkor

- az első jel csak betű lehet,
- a második jel csak számjegy,
- a harmadik jel a % jel.

Helyesen felirt egész tipusú változók: P%, X3%,AØ%, Z%

1.3.1.2. Valós változók

Ezek a változók 5 byte-os tárolóterületek helyét azonositják, képzési szabályuk u.az, mint az egész változóké, csak a 2.ill. 3. helyre nem irhatunk % jelet, és ennek következtében a változónév csak egy vagy két karakteres lehet.
Helyesen felirt valós változók: P, X3, Z, AØ

1.3.2. Szöveges változók

A változónév képzési szabálya megegyezik az egész tipusú változónév képzési szabályával, de annak jelzésére, hogy az ilyen adatok tárolóhely igénye eltér az egész változók 2 byte-os helyigényétől, a % jel helyett a ¤ jelet (valuterjel) kell használni. A szöveges változó a hozzá rendelt szöveg karakterszáma + 7 byte-os tárterületet foglal le.
Helyesen felirt szöveges változók: P¤, X3¤, Z¤, AØ¤

1.4. Kifejezések

Konstansokat és változókat műveleti jelekkel összekapcsolva kifejezéseket kapunk.
A kifejezés fogalmat általánosabb értelemben is használjuk, igy a kifejezés jelentheti csak kons-

tanst, csak változót ill. ezeknek és függvényhivatkozásoknak műveleti jelekkel összefűzött sorozatát.
A kifejezéseket kiértékelve (ha azok legalább egy műveletet tartalmaznak) valamilyen konstanst kapunk eredményül, ennek a konstansnak a tárolására megint csak meghatározott hosszuságu tárterület szükséges, igy természetes, hogy a kifejezéseket is az eddig megismert adattipusok alapján kell vizsgálni.

1.4.1. Aritmetikai kifejezések

Numerikus konstansok és aritmetikai változók aritmetikai műveleti jelekkel való összekapcsolása során keletkeznek.
Az aritmetikai műveletek és jeleik:

+ összeadás jele
- kivonás jele (legalsó billentyűsorban jobbról a 3., alsó karakter)

* szorzás jele
/ osztás jele
x x hatványozás jele (kétszer közvetlen egymás után a szorzás jele. Vigyázat, ez nem négyzetreemelés, ez a hatványozás jele)

A megismert műveleti jelekkel a

$$\frac{4 \cdot 5^3 \cdot \pi}{3}$$

képlet a következőképpen irható át aritmetikai kifejezéssé:

4 * 5 ** 3 * 3.14159/3

Látható, hogy a kifejezés csak egy sorba irható, igy a kitevőt és a nevezőt is a kifejezés sorába kell gépelni. A π karakter nem található meg a billentyűzeten, igy be sem irható, ezért számértékót kell begépelni helyette. Mivel azonban a π-t gyakran használjuk számitásainkban, az ABC 8Ø-nál lehetőséget teremtettek arra, hogy a 3.14159 szám helyett egyszerűen a PI szöveget gépeljük be, ennek hatására a gép π számértékével dolgozik majd. Vigyázzunk azonban, mert ez a lehetőség (hogy a görög betű fonetikus alakja beirható) egyetlen más esetre sem áll fenn.

Megkérdezheti, hogy miért az 5-öt emeli köbre majd a gép, és miért nem pl. 2Ø-at (4 x 5), és miért köbre emel, miért nem a 3 x 3.14159/3 hatványra.

A magyarázat a következő: a gép a műveletek fajtájától függően különböző sorrendben végzi el a müveleteket, egyszerre mindig csak egyetlen egyet. A legelsőnek végrehajtandó művelet a két szomszédos szám közötti hatványozás (attól függetlenül, hogy a művelet a kifejezés elején, belsejében vagy végén található), majd a szorzás és osztás következik, végül pedig az összeadás és kivonás végzendő el.

A műveleteknek ezt az " erősorrendjét" precedenciának nevezzük. Azonos precedenciájú műveleteknek a kifejezésben való előfordulásakor a balról-jobbra szabály érvényesül, vagyis a műveletek végrehajtása a kifejezés bal oldalától indul és onnan halad végig a kifejezésen.

Ennyi eszmefuttatás után nézzük, hogyan is számithatja ki a gép a felirt kifejezés értékót:

1. lépés: elvégzi a legmagasabb precedenciájú műveletet (ez a hatványozás) a műveleti jelet közrefogó két számmal, eredményül egyetlen szám keletkezik (5 xx 3 = 125), ezt fol-

beszámálva a kifejezés igy irható:

4 x 125 x 3.14159/3

2. lépés: a kifejezésben csak azonos precedenciájú műveletek vannak, ezért a balról jobbra szabály miatt most a legbaloldalibb szorzást (4 x125) kell elvégezni, ennek eredménye 5ØØ.
Ezzel a kifejezés jelenlegi alakja:

5ØØ x 3.14159/3

3. lépés: balról jobbra szabály, elvégzendő az 500x3.14159 művelet, eredménye 157Ø.79, a kifejezés pedig

157Ø.79/3

4. lépés: egyetlen művelet van, ennek eredménye

523. 599

A gép természetesen csak a végeredményt adja meg, a közbensőket nem.
A precedencia sorrendet a zárójel alkalmazása módositja, mert a legelőször kiértékelendő kifejezésrész a zárójelbe tett művelet (vagy műveletcsoport) lesz. A zárójelen belül természetesen változatlanul érvényesek a precedencia szabályai.

A bemutatott példában valós konstansokat használtunk.

1.4.1.1. Egész kifejezések

Egész kifejezésről akkor beszélünk, ha a kifejezés minden eleme (müveleti jeleken kivül) egész tipusú, ilyenkor az eredmény is egész tipusú lesz.

P1: $\frac{49 \times 8}{5 \times 3}$ - nak megfelelő egész kifejezés:

49% x 8% / 5 % / 3 % vagy

49% x 8% / (5% x 3%)

ennek értéke: 26% (mert 392% / 15%), de

a $\frac{49}{5} \times \frac{8}{3}$ formulának megfelelő egész kifejezés

49% / 5% x 8% / 3%

és ennek értéke: 18% (mert 49%/5% = 9% és 8%/3% = 2%)

1.4.1.2. Valós kifejezés

Ha egy aritmetikai kifejezésnek legalább egy eleme valós, akkor a kifejezés is valós és az eredmény is valós konstans lesz.

P1: $\frac{49.\times 8}{5 \times 3}$ -nak megfelelően

49x 8% / (5% x 3%)

értéke: 26.1333

1.4.2. Szöveges kifejezés

Szöveges kifejezésben csak szövegkonstansok és szöveges változók használhatók az egyetlen karakterművelet mellett.
Az értelmezett karakterművelet a konkatenáció: jele + .
A művelet eredményeként a jel két oldalán álló szöveg összekapcsolódik.

P1: "TER" + "VEZET"
eredménye: "TERVEZET"

1.5. Standard függvények

Az ABC 80 munkánk megkönnyitésére több gyakran használt függvény értékét a függvény nevének és független változójának begépelésére kiszámitja. Ezeket a függvényeket standard függvényeknek nevezzük. A fontosabb függvénynevek és funkcióik:
(X= aritmetikai kifejezés)

SIN (X), COS (X) az aritmetikai kifejezés értékét ivmértékben adott szögnek tekintve a megfelelő szögfv. értéket adja

ATN (X) a kifejezés arctg-ét adja ivmértékben ($-\frac{\pi}{2}$; $\frac{\pi}{2}$) között

EXP (X) e alapu exponenciális műveletet (e^X) végez a kifejezéssel

LOG (X) a kifejezés természetes (e) alapú logaritmusát adja (x > 0)

LOG 1Ø (X) a kifejezés tizes alapú logaritmusát adja (x > 0)

SQR (X) a kifejezés négyzetgyökét számitja (x ≥ 0)

ABS (X) a kifejezés abszolut értékét adja

RND Ø - Ø 999999 közötti véletlen számot állit elő

INT (X) az aritmetikai kifejezés egész része, (az x értékénél kisebb vagy egyenlő egész érték)

FIX (X) az aritmetikai kifejezésben benntfoglalt legnagyobb egész szám

SGN (X) előjel függvény:

$$SGN\ (X)=\begin{cases} 1 \text{ ha } x > \emptyset \\ \emptyset \text{ ha } x = \emptyset \\ -1 \text{ ha } x < \emptyset \end{cases}$$

A standard függvények kifejezésekben is elhelyezhetők, ilyenkor a precedencia sorrend úgy alakul, hogy legelőször a függvényérték számitódik ki, (ha ezekből több van, akkor a balról jobbra szabály szerint) majd a hatványozás, stb.

1.6. Direkt üzemmód

Ebben az üzemmódban a begépelt - és a gép által értett - speciális szavakat (parancsokat) az ABC-80 a sor befejezése (RETURN lenyomása) után azonnal végrehajtja.
A parancsok közül most csak egyet ismertetünk.

PRINT

A parancs végrehajtásának eredményeképpen a gép a következő sorban kezdve kiirja a PRINT szó után álló argumentumnak megfelelő karaktereket.
Az argumentum lehet (többek között):

- szövegkonstans
- numerikus kifejezés: numerikus konstansokból és műveleti jelekből álló értelmezhető sorozat (pl: 2 + 3)
- az előző kettő ismétlése vagy kombinációja

A parancs müködésének ellenőrzéséhez billentyűz-

ze be a következő sort:
(cursor a sor elején legyen)

PRINT "ABC - 8Ø ␣ A ␣ LEGJOBB" <RET>

A záró idézőjel után irt <RET> azt jelenti, hogy a RETURN billentyű lenyomásával a sort itt le kell zárni. A ␣ jellel a szóközt jelöltük. Ha minden jól sikerült, a következő sorban (a sor első pozíciójától) az ABC-8Ø ␣ A ␣ LEGJOBB
szöveget kell kapnia.
A kifejezések felirását, értékük kiszámitását direkt üzemmódban gyakorolhatjuk.
Mielőtt begépeli a következő parancsokat, gondolja meg, hogy mi is lesz majd az eredmény. Ha a gép azt adja eredményül amit várt, akkor megértette az eddigieket, de ha nem, akkor jobban odafigyelve próbálja átismételni az anyagot.

```
PRINT    2 * 2
PRINT   -2 ** 2
PRINT    (-2) ** 2
PRINT    3 * 1/3
PRINT    3 * (1/3)
PRINT    3 % * (1% /3%)
PRINT    "A = "; 1E6 * 25 ** (1/2)
PRINT    "A=", 1E-6 * 25 **  1/2
PRINT    "Y =" ; 8 * 4/3/6
PRINT    "Y =" ; 8 /3 * 4/6
```

Az utolsó 4 PRINT parancs argumentumában több kinyomtatandó elem volt, pl. a legutolsóban a "Y=" szövegkonstans és a 8/3 * 4/6 numerikus kifejezés. Attól függően, hogy az elemeket milyen jel /vessző vagy pontosvessző/ választotta el, különböző kiirási képet kaptunk. Vessző elválasztójel esetén az ABC 8Ø az elemeket alkotó jeleket 14 poziciót tar-

talmazó oszlopok elején kezdi kiirni, pontosvesszõ esetén pedig közvetlenül egymás után. Jól látható ez a szabály a köv. példákban.

```
PRINT    "A", "A", "A"
eredménye:   A           A           A
                  12 szóköz   12 szóköz

PRINT    "A"; "A"; "A"
eredménye:
AAA
```

1.7. Programkezelés

Az ABC 8Ø -at általában indirekt üzemmódban használjuk, mert többnyire nemcsak egyedülen parancs, hanem utasitások egész sorozatának automatikus végrehajtását kivánjuk a géptől. Indirekt üzemmódban a begépelt utasitásokat a gép tárolja, és csak egy külön futtató parancs beadására kezdi el azok végrehajtását. A tárolt utasitások sorozatát nevezzük programnak.

A programokat tárolhatjuk a gép belső tárolójában, de mágneskazettán is megőrizhetjük azokat a későbbi ismételt felhasználások céljára. Az ABC 8Ø - hoz mellékelt DEMO nevü kazettán több demonstrációs, bemutató program található. A géppel való további ismerkedéshez használjuk ezt a kazettát, és a program kazettára való kimentéséhez vegyünk egy üres kazettát is.

1.7.1. Program betöltése kazettáról

Nyomja meg a kazettás egység STOP/EJECT billentyűjét, ekkor a kazettafedél felnyilik és a kazetta behelyezhető, majd a fedelet csukja le. Ha a sza-

lag nem az elején van (ilyenkor felülről nézve a jobb oldali orsónak üresnek kell lenni), tekercselje vissza a szalagot a bal szélső (◀◀) gombbal.
Alaphelyzetben nyomja meg a RESET billentyűt (a számláló mellett,), ezzel nullázza a számlálót.
Az első beolvasandó program a kazetta elején található, kb. a 10-es számláló állásnál és kb 90-esnél fejeződik be. Beolvastatásához

- nyomja le a kazettás egységen a PLAY billentyűt
- gépelje be a klaviatúrán a

  LOAD CAS :<RET> (jelentése: töltsd be kazettáról a szalagon sorban következő programot)

  parancsot.

A szalag futása csak a RETURN gomb lenyomásakor indul meg. A program tényleges beolvasásakor a monitor hangszórójából az átvitelre jellemző kü - lönleges, sipoló-recsegő hangokat hallhat.
Ha az átvitel nem sikerül, a gép a képernyőre az

ERR 21 hibaüzenetet irja.

Kisérelje meg újra a betöltő parancs begépelését.
Ha az átvitel megkezdődik, a gép kiirja a megtalált és éppen betöltés alatt álló program nevét (a kirásban aláhúzással jelöljük) a következőképpen:

FOUND DEMO BAC

A program végére jutva jelenik meg ismét a képernyőn az ABC 8Ø felirat, ekkor a program a gép tárolójában van.

1.7.2. A program elinditása, (futtatása)

A tárban lévő program elinditását a RUN (fuss) paranos begépelésével tehetjük meg, azaz begépelendő:

RUN RET

A RETURN megnyomása után a program elindul, majd automatikusan ujrainditja önmagát. Ha már nem kivánja többször lefuttatni a programot, állitsa le annak futását. (Ld. 1.1.2. pont)

1.7.3. Program felvétele kazettára

Kazettára azokat a programokat lehet kimenteni, melyek az ABC-80 tárolójában vannak - egyszerre persze csak egy program lehet a tárban.
A mentéshez tegyük be azt a kazettát, amire a programot fel akarjuk venni, majd a számláló segitségével állitsuk a szalagot abba a pozicióba, amibe a felirandó program nem okoz problémát (pl. nem ir felül olyan szalagrészt, ahol még később is szükséges program található). A szalag beállitása után

- nyomja le egyszerre a PLAY és REC gombokat és

- gépelje be a klaviaturán a
  SAVE programnév RET (jelentése: mentsd el kazettára a gépben tárolt programot az adott programnévvel)

A SAVE után gépelendő programnevet a felhasználó találja ki célszerűen ugy, hogy a név a programra vo-

natkozóan érdemi információt tartalmazzon. A DEMO név pl. a demonstrációs (bemutató) program neve.
A kazettára való mentés akkor fejeződik be, ha a gép ismét kiirja az ABC 8ø feliratot.
A mentendő program neve legfeljebb 8 karaktert tartalmazhat.

2. Az ABC 8Ø BASIC alaputasitásai

A BASIC az egyik legegyszerűbb, nagy népszerűségnek és felhasználhatóságnak örvendő programozási nyelv, melynek első változatát 1963-ban az USA-ban a magyar származású John G.Kemény vezetésével dolgozták ki. A BASIC elnevezés betűrövidités, a Beginers All-purpose Symbolic Instruction Code (kezdők általános szimbólikus utasitás kódja) szavak kezdőbetűiből áll össze.

Ma már igen sokféle számitógépen használhatjuk a BASIC programozási nyelvet, sajnos, szinte minden gép rendelkezik olyan speciális lehetőségekkel, melyek más gépek BASIC rendszerében nem találhatók meg. Ebben a fejezetben azokat a BASIC utasitásokat ismertetjük, melyek általában minden gép BASIC rendszerében megtalálhatók, és nem sok eltérést mutatnak a tárgyalt ABC 8Ø BASIC változathoz viszonyitva.

2.1. BASIC programok irásának formai szabályai

A BASIC program utasitásokból áll. A program gépbe való bebillentyűzése során ezeket, a különböző müveleteket kiváltó utasitásokat a gép tárolójába visszük be, majd a program végrehajtásakor a számitógép az utasitásokat egymás után automatikusan előveszi, értelmezi és az előirt müveleteket elvégzi.

Általában a jó áttekinthetőség végett egy sorba egy BASIC utasitást irunk. Az ABC 8Ø BASIC megengedi egy sorban több utasitás szerepeltetését is, ilyen esetekben a szomszédos utasitásokat kettősponttal kell elválasztani egymástól.

Az utasitások végrehajtási sorrendjének egyértelmüvé tételére a BASIC programsorokat cimkével (sorszámmal) kell kezdeni. A cimkék pozitiv egész számok lehetnek, értékük az 1,65535 intervallumban változhat. A gép az utasitásokat a cimkék növekvő sorrendjében (tehát nem a begépelési sorrendben) hajtja végre - hacsak más-

képp nem rendelkezünk (ld. 2.6, 2.7 pont). A cimkéknek nem kell egyesével növekvően követniük egymást, bármilyen hézagok megengedhetők a cimkék között a nulla kivételével, azaz két azonos cimkéjű BASIC sor nem szerepelhet egy programban. Szokásos a cimkéket tizesével növekvően képezni, hogy az esetleg szükséges programjavitásoknál a meglévő sorok közé újabb, javitó sorokat iktathassunk be.

Minden BASIC sort sorvégjellel kell lezárni, ez az ABC-8Ø-nál a RETURN (kocsi vissza, soremelés) karakter. Vigyázat! Ez a szó egyetlen billentyűt jelent (ld.1.1.2. pont).

A BASIC utasitások általános felépitése:

cimke Utasitás kulcsszó operandus sorvégjel

Az utasitás kulcsszó azt mondja meg a gépnek, hogy mit kell csinálnia, milyen müveletet kell elvégeznie, az operandus pedig azt mutatja, hogy mivel vagy min kell az adott müveletet végrehajtani. Vannak olyan utasitások is amelyekben több kulcsszó ill. operandus szerepelhet, néhány utasitásban pedig operandusra nincs is szükség.

Az utasitás kulcsszavak angol nyelvű szavak vagy azok rövidítései. Ezeket általában nagy betűkkel kell irni, az ABC 8Ø azonban a kis betűvel irt kulcsszavakat is képes értelmezni.

Az ABC 8Ø monitorán egy sorban 4Ø karakter helyezhető el, azonban a gép számára egy sort 12Ø karakter jelent, azaz egy valóságos gépi sor megjelenitése a képernyőn három részre darabolva is történhet. A sornak természetesen nem kell kötelezően 12Ø jelet tartalmaznia, a sor minden esetben a sor kezdetétől a sorvégjelig terjed. Ha egy képernyősorban nem fér el az utasitás, akkor mindenféle elválasztójel és karakterismétlés nélkül folytatjuk a begépelést a következő sorba.

Ha a sor lezárása előtt észrevettük, hogy a begépelt utasitásban hiba van, akkor a ← billentyű segitségével

visszavisszük a cursort az utolsó jó karakterhez (közben az összes rossznak minősitett jelet törli a gép), és innen folytatva az utasitásbeirást a helyesnek tartott karaktereket gépeljük be.

Ha a sor lezárása után a gép hibaüzenetet ad, akkor az utasitást javitani kell, legegyszerűbb módon úgy, hogy a rossz utasitás cimkéjét újra begépeljük, majd a jónak tartott utasitást irjuk be. További javitási lehetőségeket a 3.5. pontban találhatunk.

2.2. Értékadás a LET utasitás

A LET értékadó utasitással változónak érték adható. Az utasitásban az egyenlőségjel bal oldalán helyezkedik el az a változó, melyhez értéket rendelünk a jobb oldalon pedig egy kifejezés.
Az utasitás végrehajtásakor a rendszer kiértékeli a jobb oldalt és annak értékével egyenlővé teszi az egyenlőségjel bal oldalán levő változót.
A LET értékadó utasitásban a LET szó opcionális.

Példák értékadásra:

```
40 LET J=3
50 J=3
```

```
10 X=A*L-Z*Y
20 A$=B$+C$
```

```
10 C$='ABCDEF'
20 A$='12345'
```

2.3. Adatbevitel- az INPUT utasitás

Az INPUT utasitás lehetővé teszi, hogy a tárban tárolt program futtatása során adatokat vigyünk be a gépbe. Az utasitás végrehajtásakor a program futása fel-

függesztődik, egy kérdőjel jelenik meg a képernyőn és rendszer adat bevitelre vár a billentyűzetről.

Pl. a

```
10  INPUT  A,B,C2
```

utasitás végrehajtásakor a képernyőn megjelenik a kérdőjel és a rendszer vár az adatok billentyűzetről történő bevitelére.

A beirt adatokat a rendszer az INPUT utasitásban megadott sorrendben rendeli a változónevekhez (ezért ügyelni kell a különböző tipusú adatok beadásánál, hogy a hozzárendelés megfelelő legyen!). Több adat bevitele esetén azokat egymástól vesszővel vagy RETURN billentyűvel kell elválasztani, a bevitelt a RETURN billentyű lenyomásával zárjuk le.
A beirt adatokon a RETURN billentyű lenyomását megelőzően módositásokat lehet végezni (← billentyű).
Ha kevesebb adatot irunk be vesszővel elválasztva, mint amennyi változónév az INPUT utasitásban van, lenyomjuk a RETURN billentyűt, akkor a következő sor elején újból megjelenik a kérdőjel és a rendszer vár a további adatbevitelre.
Ha a bevitt adatok száma több, mint az INPUT lista elemeinek a száma, akkor a rendszer a felesleges adatokat (az utoljára begépelteket) figyelmen kivül hagyja.

Példák az utasitás alkalmazására:

1. Példa

```
10 INPUT  X
 .                (Ezzel a paranccsal inditjuk el a
 .                program végrehajtását.)
 .
RUN
 ? 15.6    RETURN
```

2.Példa

```
10 INPUT X,Y
RUN
?17,23
```

3.Példa

```
1Ø INPUT A,B$
RUN
?17.2,'KANDO'
```

4.Példa

```
1Ø INPUT A$,B$
RUN
?'KANDO','FÖISKOLA'
```

Megjegyzés: A szöveg változókhoz rendelt karakterfüzérek idézőjelek nélkül is megadhatók.

2.4. Nyomtatás - a PRINT utasitás

A PRINT utasitás segitségével megjelenithető tetszőleges szöveg; változók, kifejezések, konstansok értékei irathatók ki, üres sorokkal szétválasztható a kiirt információ, valamint a kiiratási formátum beállitható.

A PRINT utasitás parancsként is kiadható.
A PRINT szó az ABC 8Ø gépnél ;-vel (pontosvesszővel) helyettesithető a programsor beirásakor.

A PRINT kulcsszót (vagy ;-t) követő listában lévő elemeket (melyek tehát konstansok, változók, kifejezések lehetnek - numerikus vagy szöveg tipusúak) elválasztó jelekkel kell egymástól elválasztani. Ezek

a lehetséges elválasztó jelek: , és ; . A vesszőnek és a pontosvesszőnek meghatározott kiírásvezérlő szerepe van. Ha a listaelemek között vessző található, akkor a vessző után álló elemet (vagy értékét) a következő 13-mal osztható poziciótól kezdődően irja majd ki a gép a képernyőre. A 4Ø pozicios sor miatt egy sorban három nyomtatási oszlop alakitható ki, az oszlopok első poziciói a Ø, 13, 26 poziciók. Ha a nyomtatandó elemek száma háromnál nagyobb és vesszőt használunk elválasztó jelként, akkor a 3. elem utáni elemeket a következő sorba, ugyancsak oszloposan irja ki az ABC 8Ø.

Pl.:

```
        :

110 PRINT 'OSSZESEN:',S1,S2

        :
```

Ennek a programsornak a végrehajtásakor (ha S1 tartalma 12Ø és S2-é 1918.65) a köv. sor jelenik meg:

OSSZESEN: 12Ø 1918.65

Ha a vessző után újabb listaelem nem áll, akkor a következő PRINT utasitás első listaelemének kiirása kezdődik majd ugyanazon kiirási sor következő oszlopában.

Pl.:

```
        :

120 PRINT 'OSSZESEN:',
130 PRINT S1,S2

        :
```

A programrészlet végrehajtása után az előbb látott kiirási kép jelenik meg.

A pontosvessző elválasztó karakter hatására a ki-

nyomtatandó elemek közvetlenül egymás után jelennek meg. Ha a PRINT utasitás listáját zárja pontosvessző, akkor a következő PRINT első eleme közvetlenül a már megkezdett kiirási sorban az utoljára kiirt adatot követően jelenik meg.
A kiirandó érték kiiratási helyének kezdetét kijelölhetjük a PRINT utasitásban elhelyezett TAB (kifejezés) alkalmazásával, ahol a kifejezéssel adjuk meg az első karakter pozicióját.

Pl.:

```
8Ø PRINT TAB(18);"MEGOLDAS"
```

utasitás végrehajtásakor a nyomtatási kép:

MEGOLDAS

A PRINT utasitásban a CUR (kifejezés1, kifejezés2) megadásával a cursort a képernyő adott sorára és oszlopára állithatjuk be.

0≤kifejezés1	≤23	a sorokra vonatkozóan
0≤kifejezés2	≤39	az oszlopokra vonatkozóan

1.Példa

```
1Ø A=1234 : B=-5678
2Ø PRINT "ABC";"DEF"
3Ø PRINT A;B
RUN
ABCDEF
 1234-5678
```

2.Példa

```
1Ø A=1234 : B=-5678
2Ø PRINT "ABC";"DEF";
3Ø PRINT A;B
RUN
ABCDEF  1234-5678
```

3.Példa

```
10 A=1234 : B=-5678
20 PRINT "ABC";"DEF",
30 PRINT A;B
RUN
ABCDEF          1234 -5678
```

2.5. Programzárás - STOP, END utasitás

A BASIC program végrehajtását a STOP, illetve END utasitás segitségével lehet megállitani. E két utasitás a programban bárhol alkalmazható (nemcsak a program legmagasabb sorszámú soraként).
A STOP ill. END kulcsszó után nem áll operandus.
A két utasitás hatása egymástól a következőkben tér el:

- A STOP utasitás végrehajtása során a rendszer - miután a BASIC program végrehajtását leállitotta - a képernyőn a következő üzenetet jeleniti meg:

STOP LINE az utasitás sorszáma

(az üzenet megjelenésével egy időben rövid hangjelzés hallható).
A STOP utasitás végrehajtása után a rendszer parancs módba vált át, és a felhasználó a program változóinak aktuális értékeit kiirattathatja.

- Az END utasitás végrehajtása során a rendszer - miután a BASIC program végrehajtását leállitotta - a képernyőn a következő üzenetet jeleniti meg:

ABC8Ø (hangjelzés most nem hallható)

Az END utasitás hatására - ellentétben a STOP utasitás hatásával - a rendszer a nyitott file-okat

lezárja és valamennyi változó értékét törli, igy azoknak az END utasitás végrehajtása előtti értéke - miután a rendszer parancs módba váltott - nem kérdezhető le.

Ha a programban egyetlen STOP vagy END utasitás sincs, akkor a rendszer a legmagasabb sorszámú sor végrehajtása után a programvégrehajtást úgy állitja meg, mintha egy END utasitást hajtott volna végre.

2.6. Feltételes vezérlésátadás - IF - THEN - ELSE utasitás

Az IF utasitás segitségével feltételes vezérlésátadás valósitható meg.

Az IF utasitás-kulcsszó után és a THEN kulcsszó előtt egy összehasonlitó kifejezés áll. Az összehasonlitó kifejezésben két kifejezés közötti reláció teljesülését vizsgáljuk.

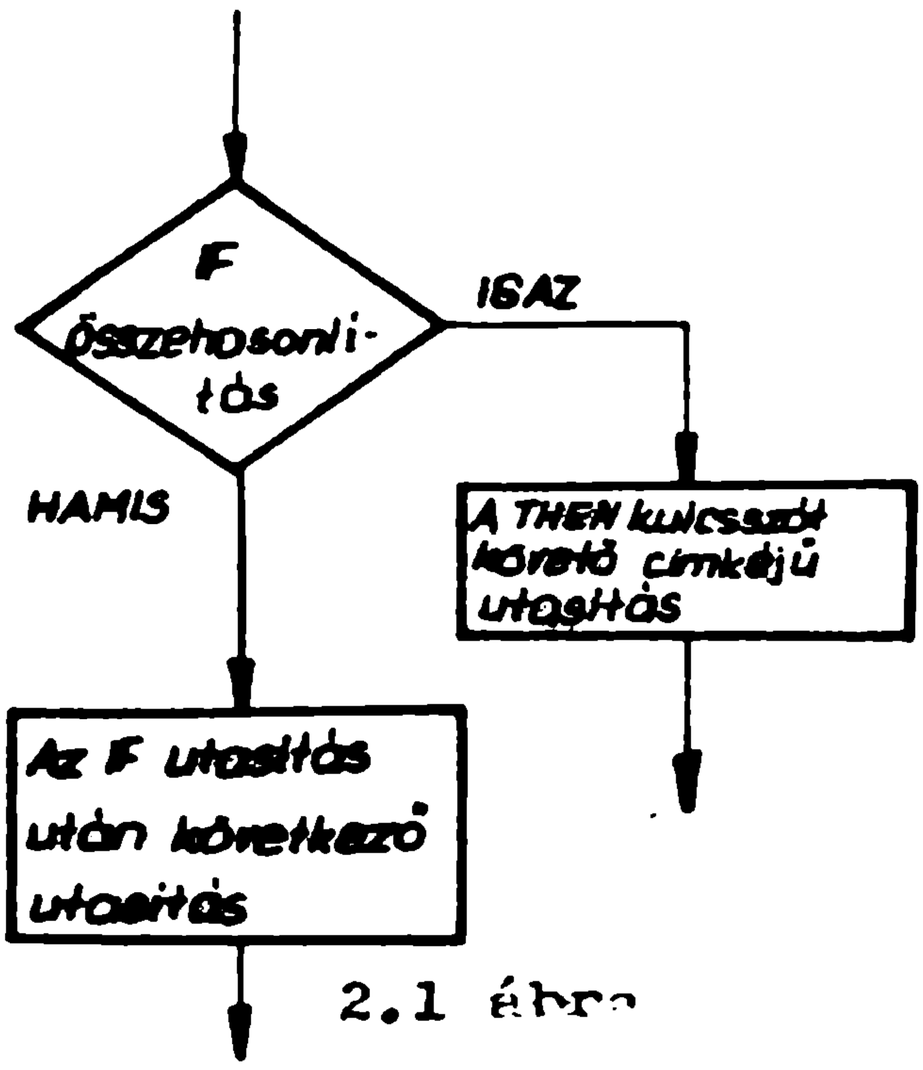
A vizsgálható relációkat a 2.1. táblázatban foglaljuk össze:

Reláció	Matematikai forma	BASIC forma
egyenlő	=	=
nem egyenlő	≠	<>
kisebb	<	<
kisebb vagy egyenlő	≦	<=
nagyobb	>	>
nagyobb vagy egyenlő	≧	>=

2.1. táblázat

Az összehasonlitó kifejezés értéke vagy IGAZ (IGEN) vagy HAMIS (NEM). Ha a két kifejezés közé irt reláció teljesül, akkor az összehasonlitó kifejezés értéke IGAZ, ha a reláció nem teljesül; akkor HAMIS.

Az utasitás végrehajtása során a rendszer először logikailag kiértékeli az összehasonlitó kifejezést. Amennyiben az abban kijelölt reláció teljesül - a kifejezés logikai értéke = IGAZ - a program végrehajtása a THEN kulcsszót követő utasitás sorszám által megjelölt sortól folytatódik. Abban az esetben azonban, ha a kijelölt reláció nem teljesül - a kifejezés logikai értéke =HAMIS - a program végrehajtása az IF utasitást követő utasitástól folytatódik. A feltételes vezérlésátadást a 2.1. ábrán szemléltetjük.



2.1 ábra

1. Példa

```
  5  X=3.1 : Y=2.
  9  IF X=3 THEN 24
 12  IF INT(X)=3 THEN 55

 55  IF Y<X*X THEN 24
 57  IF Y>X THEN 2.1

2..3  IF 2*X<Y THEN 22
21.   STOP
22.   IF X+Y<=1. THEN 3..
2..   PRINT Y-X
24    STOP

3..   PRINT X+Y : STOP
```

A 9 sorban lévő utasitásban lévő reláció nem teljesül, mivel 3.Ø1≠3, ezért a végrehajtás a 12 sortól folytatódik. Ebben a sorban viszont a reláció már teljesül, mert INT(3.Ø1)=3 és igy a program végrehajtása az 55 sortól folytatódik. Az 55 sorban lévő reláció HAMIS, ezért a rendszer áttér a 6Ø utasitássor végrehajtására, amelyben lévő reláció IGAZ, mivel Y > X, igy a végrehajtás a 2ØØ sortól folytatódik. A 2ØØ sorban a reláció ismét IGAZ, igy most a 22Ø sorszámú sor hajtódik végre. Ebben a sorban a reláció értéke HAMIS, ezért a program végrehajtása a 23Ø sortól folytatódik, majd megáll a 24Ø sornál.
Mint látható, a THEN kulcsszó után egy utasitás sorszám adható meg, amelyre a vezérlés a reláció teljesülése esetén átadódik. Ezen utasitás cimke helyén azonban egy-vagy egymástól :-al elválasztva több - BASIC utasitás is szerepeltethető:

2.Példa

```
14 INPUT F1
16 IF F1=RND(17 THEN F2=F1*8 : PRINT F2
18 PRINT F1
20 STOP
```

A példában a rendszer beolvassa az F1 változó értékét, majd megvizsgálja, hogy a változó aktuális értékével számolva a reláció teljesül-e. Amennyiben a reláció logikai értéke =IGAZ, végrehajtja a THEN kulcsszót követő két BASIC utasitást, majd rátér a 18 sor végrehajtására, kinyomtatja F2, majd F1 értékét. Ha a reláció nem teljesül, a program azonnal a 18 sortól folytatódik,csak F1-et nyomtat. A THEN kulcsszót követően - az un. THEN-ágban - tetszőleges BASIC utasitás - akár újabb IF is - szerepelhet
Ha a THEN kulcsszó után nem ugrócimet, hanem utasitást (utasitásokat) irunk, akkor figyelni kell arra, hogy ezen utasitások végrehajtása után a vezérlés

ugyanarra a sorra kerülhet, amelyre az IF utasitásban levő reláció nem teljesülése után adódna. Ha erre nincs szükség, akkor a THEN utáni utolsó utasitásnak vezérlésátadásnak kell lennie.

Az eddig leirtakon túlmenően lehetőség van arra is, hogy az IF utasitásban arra az esetre is kijelöljünk utasitásokat, ha a reláció nem teljesül.

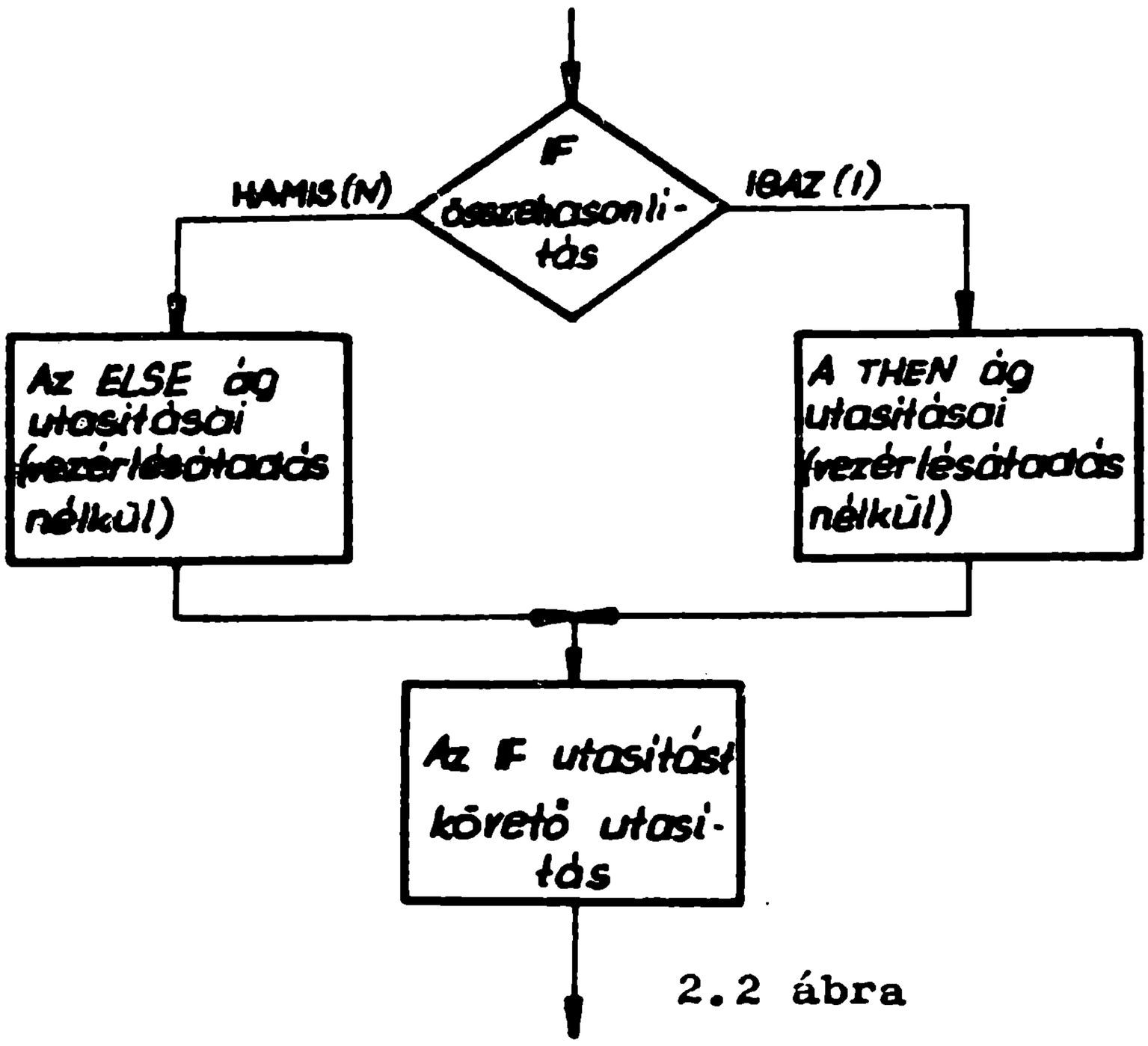
Ekkor az un. ELSE ágat kell alkalmaznunk:

3.Példa

```
14 INPUT F1
16 IF F1=6<=17 THEN F2=F1*8 : PRINT F2 ELSE PRINT F1
26 STOP
```

Ha a 16 sorban lévő IF utasitásbeli reláció teljesül, a rendszer a THEN ágban lévő utasitásokat hajtja végre, mig ha a reláció nem teljesül az ELSE ág hajtódik végre. Mint látható, ez utóbbi példa az előzőkhöz hasonló feladatot hajt végre, csak most hiányzik a 18 utasitás, amelynek szerepét az ELSE ág vette át. Most a program a reláció teljesülése esetén csak F2-őt nyomtatja ki.

Az ELSE ágban - THEN ághoz hasonlóan - egynél több tetszőleges BASIC utasitást szerepeltethetünk, amennyiben azokat egymástól a : segitségével elválasztjuk. Az IF-THEN-ELSE szemléltetésére közöljük a 2.2. ábrát:



2.2 ábra

Ezen túlmenően lehetséges, hogy az ELSE ágban mindössze egyetlen utasitás cimkét szerepeltessünk azonos módon, mint ahogy azt a THEN ág esetén már ismertettük.

4.Példa

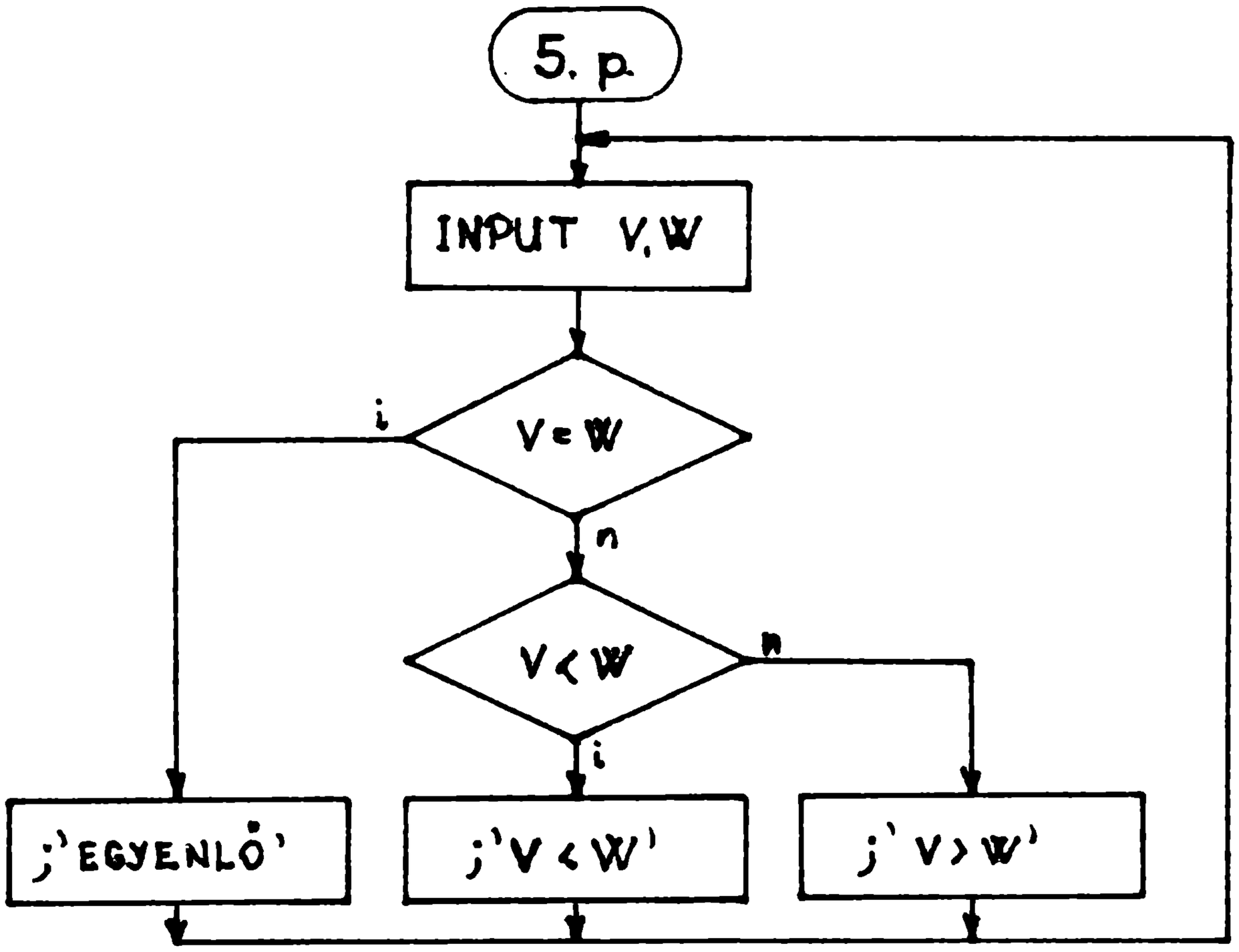
```
 20 INPUT K$
 40 IF K$='IGEN' THEN 80 ELSE 100
 60 STOP
 80 PRINT 'IGEN' : GOTO 60
100 PRINT 'NEM' : GOTO 60
```

Példa egymásba ágyazott IF utasitások alkalmazására:

5.Példa

```
10 INPUT V
20 IF V=0 THEN PRINT 'EGYENLŐ' ELSE IF V<0 THEN PRINT
   'V<0' ELSE PRINT 'V>0'
30 GOTO 10
```

Mint látható a 2ø utasitássor egy IF utasitást tartalmaz, melynek ELSE ágában egy újabb IF utasitás helyezkedik el. Ennek az egyszerű mintapéldának a folyamatábrája a következő: (ld. 2.3. ábra)



2.3. ábra.

Mint már emlitettük, akár a THEN, akár az ELSE ágban több, egymástól :-tal elválasztott utasitás is szerepelhet. Az egyedüli korlát, hogy a BASIC utasitássor teljes hossza nem haladhatja meg a 12ø karaktert.

2.7. Feltétel nélküli vezérlésátadás - GOTO utasitás

A GOTO utasitás segitségével megváltoztathatjuk a programvégrehajtás normális sorrendjét. Az utasitás hatására a program soronkövetkező utasitása helyett a GOTO utasitásban meghatározott sorszámú sortól folytatódik a program végrehajtása.

1. Példa

```
11  A=1
21  IF A*A<1000 THEN 51
31  PRINT A,A*A
41  STOP
51  A=A+1
61  GOTO 21
```

Az 1. példában egy programhurkot láthatunk. Mindaddig, amig az A*A szorzat kisebb, mint 1000, az A változó értékét a program az 51 sorszámú sorban megnöveli 1-el, majd ezután a GOTO utasitás - 61-es sorszámú sor - hajtódik végre, melynek hatására a vezérlés visszatér a 21-es sorszámú sorra. A program egyébként kiirja azt az egész számot, melynek négyzete éppen nagyobb ezernél, és még ezt a négyzetszámot is kinyomtatja.

2. Példa

```
110 X=1
115 Z=X+Z
120 PRINT X,Z/X
125 X=X+1
130 GOTO 115
```

A 2. példában a GOTO utasitás felhasználásával egy programhurkot hoztunk létre a 115-ös és 130-as sorszámú sorok között. Ez egy un. 'zárt hurok' - másnéven végtelen ciklus -, mivel a hurok nem tartalmaz olyan utasitást, melynek hatására a hurokból kiléphetne a program végrehajtása. Igy, ha meg akarjuk állitani ezt a programot, a CONTROL-C billentyűt kell megnyomni.

A GOTO utasitásokban álló sorszámoknak konstansoknak kell lenniük; például a 15 GOTO P alakú utasitás nem alkalmazható. Továbbá a GOTO utasitásban álló sorszámmal jelölt sornak szerepelnie kell a programban, mert ha nem, a végrehajtáskor hibajelzés keletkezik.

2.8. Megjegyzések elhelyezése - REM utasitás

A REM (Remark = Megjegyzés) utasitás csupán a programozó számára szóló emlékeztető vagy megjegyzés. Ezt az utasitást a gép nem hajtja végre, azonban az utasitás megjelenik a program utasitások között a program listázásakor. A REM kulcsszót a karakterek tetszőleges sorozata követheti, az egyetlen megkötés az, hogy az utasitássor maximális hossza 12Ø karakter lehet.

Példák

```
13 REM EZ CSUPAN EGY MEGJEGYZES
23 REM---ITT AKARMI MEGJELENHET
33 REM:AYZABCD12+-*/
43 REM....) <=-REM UTASITAS
```

2.9. Ciklusszervezés FOR ... NEXT utasitás

A FOR és NEXT utasitások segitségével a BASIC programban ismételten végrehajtandó utasitáscsoportokat - ciklusokat - hozhatunk létre.
A programban a FOR és NEXT utasitások között elhelyezkedő utasitások összességét ciklusmagnak nevezzük. A FOR utasitás segitségével határozhatjuk meg azt, hogy a ciklusmagban elhelyezkedő utasitásokat a rendszer egymás után hányszor hajtsa végre.

Például:

```
10 FOR I=1 TO 5
     .
     .
     .
50 NEXT I
60   .
     .
```

A példabeli ciklus magja (a három ponttal jelölt uta-

sitások) egymás után ötször fog végrehajtódni, miközben I (a ciklusváltozó) értéke rendre felveszi az 1,2, 3,4 és 5 értéket.
A ciklusmag minden egyes végrehajtása után a NEXT utasitást is végrehajtja a rendszer, melynek során a ciklusváltozó (a példában I) értékét 1-gyel megnöveli. Mindaddig, amig a ciklusváltozó értéke nem nagyobb a FOR utasitásban meghatározott végértéknél (a példában ötnél) a ciklusmagot a rendszer ismételten végrehajtja. Amikor azonban a ciklusváltozó értéke túlhaladta a végértéket (a példában ez akkor következik be, amikor I egyenlő lesz hattal) a program végrehajtása a NEXT utasitást követő utasitástól (a példában a 6Ø-as sorszámú utasitás) folytatódik.
Annak szemléltetésére, hogy a ciklusszervező utasitás alkalmazása a feltételes vezérlésátadó utasitáshoz képest milyen előnyökkel jár, tekintsük a következő két egyszerű programot, amelyek logikailag azonos feladatot látnak el, nevezetesen sorban megjelenitik az 1-1ØØ közötti egész számokat (az 1-et és a 1ØØ-at is):

1. Példa

IF használatával

```
1Ø N=1
2Ø IF N > 1ØØ THEN 6Ø
3Ø PRINT N;
4Ø N=N+1
5Ø GOTO 2Ø
6Ø END
```

2. Példa

FOR...NEXT használatával

```
1Ø FOR  N=1 TO 1ØØ
2Ø PRINT N;
3Ø NEXT N
4Ø END
```

Amint ez a két példa összehasonlitásából látható, ciklusszervező utasitás alkalmazásával a program áttekinthetőbb, kevesebb munkával begépelhető, ezenkivül a gép tárolójában kisebb helyet igényel és végrehajtása gyorsabb, mintha feltételes vezérlésátadó utasitást alkalmaz tunk volna.

3.Példa

```
11 Z=0
21 FOR P=90 TO 100
31 Z=Z+P
41 NEXT P
51 PRINT 'AZ OSSZEG=';Z
61 END
```

A 3. példaprogram a 9Ø és 1ØØ közti egész számok összegét határozza meg, majd az összeget megjeleniti. Mint a példából látható, a ciklusváltozó kezdőértékének nem feltétlenül az 1 értékkel kell egyenlőnek lennie.

4.Példa

```
32 Z=0
42 INPUT M,N
52 FOR P=M TO N
62 Z=Z+P
72 NEXT P
82 PRINT 'AZ OSSZEG=';Z
91 END
```

A 4. példaprogram hasonló a 3. példához, azonban most a ciklusváltozó kezdő és végértékét a 42 sorszámú sorban lévő INPUT utasitás végrehajtása során az M és N változókhoz rendelt aktuális értékek határozzák meg.

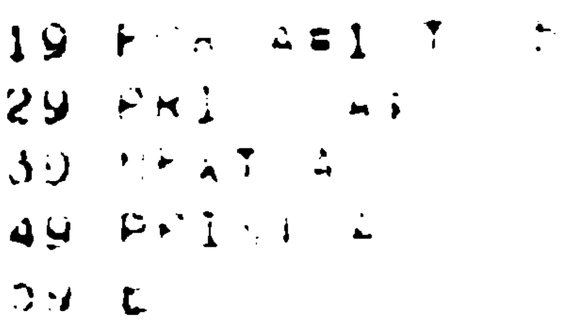
. Ha M=3 és N=5, a 3,4 és 5 összege lesz az eredmény.

. Ha M=3.1 és N=5.1, a 3.1,4.1 és 5.1 összege lesz az eredmény.

. Ha M=3.1 és N=5, a 3.1 és 4.1 összege lesz az eredmény.

Abban az esetben, ha M értéke nagyobb az N értékénél, a ciklus egyetlen egyszer sem fog lefutni, és az eredmény egyenlő lesz 0-val.

5.Példa



Mint ahogy az az 5. példaprogramban látható, a ciklusváltozó értéke a ciklus lejárta után (amikor a ciklusváltozó aktuális értéke meghaladta a végértéket) nem vész el, az a későbbiekben felhasználható. Igy, ha a példabeli programot lefuttatjuk, a képernyőn a következő számok jelennek meg: 1 2 3 4 5 6

Mint látható, a ciklus lefutása után a korábbi ciklusváltozó - amely mostmár mint egy közönséges változó alkalmazható - aktuális értéke 1-gyel nagyobb a végértéknél. Általánosságban is igaz, hogy a ciklusváltozó aktuális értéke a ciklus lefutása után nagyobb, mint a ciklusban megadott végérték. Néhány programban ez a sajátosság kisebb problémát okoz. A következő 6. példa erre mutat példát, és a probléma megoldásának módszereit is vázolja.

6.Példa

```
15 Z=0
20 INPUT V
25 FOR A=1 TO V
30 Z=Z+A
35 NEXT A
```

vagy

```
40 PRINT "AZ ATLAG=";Z/(A-1)
40 PRINT "AZ ATLAG=";Z/V
50 END
```

A példabeli program az 1-től V-ig terjedő egész számok átlagát képzi.
Mivel azonban a ciklus lejártakor - az utoljára végrehajtott növelés miatt - az A ciklusváltozó értéke 1-gyel nagyobb az összegzésben részt vett értékek számánál, a helyes átlagérték meghatározása érdekében az osztást a ciklusváltozó aktuális értéke "-1"-gyel kell végrehajtani (lsd. a 40-es sort), kompenzálva ezzel az utolsó növelés hatását. Egy lehetséges másik megoldás szerint nem a ciklusváltozó aktuális értékét, hanem a ciklus végértékét - amely a fenti példában egyenlő az összegzésben részt vett értékek számával - alkalmazzuk osztóként (lsd. a második 40-es sort).

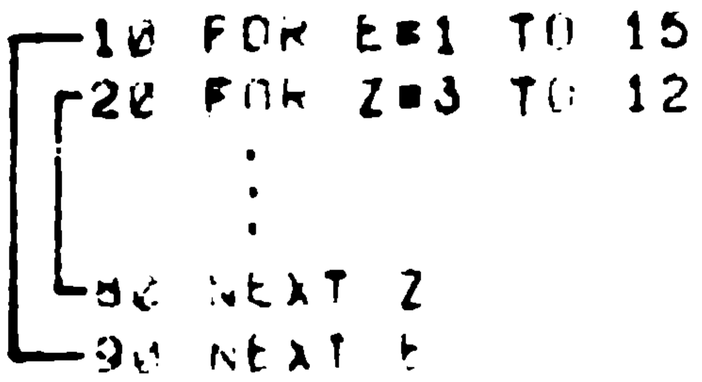
7.Példa

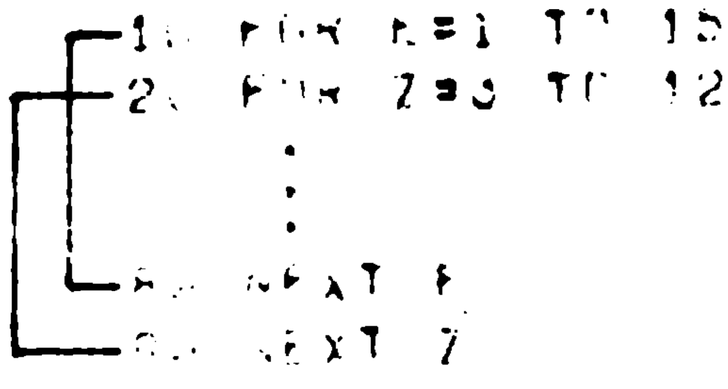
```
1 FOR G=1 TO 100
2 IF G*G>=529 THEN 5
3 NEXT G
4 STOP
5 PRINT G,G*G
6 END
```

A 7. példa azt szemlélteti, hogy a ciklusból annak lejárta előtt is ki lehet lépni. Amikor a fenti program futása befejeződik, a képernyőn a következő számok láthatók: 23 529

Egy ciklus magjában elhelyezkedhet egy másik ciklus is. Ezt nevezzük a ciklusok egymásba ágyazásának, vagy egymásba skatulyázásának. E lehetõség alkalmazásakor azonban figyelembe kell venni, hogy a belsõ ciklusnak teljes egészében benne kell lennie a külsõ ciklus magjában.

Helyes ciklus egymásba épités Hibás

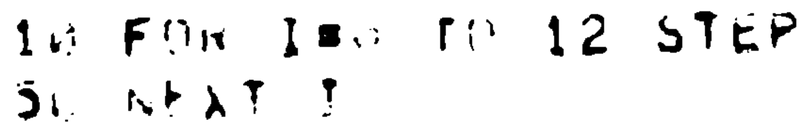




Figyelem: Minden FOR utasitáshoz kell, hogy tartozzék egy - NEXT utasitás. Ellenkező esetben a rendszer hibát fog jelezni.

Számos esetben hasznos lenne, ha a ciklusváltozó aktuális értéke a ciklusmag egymást követõ végrehajtásai során nem 1-gyel növekedne. A BASIC programozási nyelv erre lehetõséget ad a ciklus paramétereinek megadása során.

Pl.:



A FOR utasitás sorban a STEP kulcsszó szerepeltetésével elérhető, hogy a ciklusmag minden egyes végrehajtása után a ciklus változóját a rendszer a STEP kulcsszót követő értékkel (a lépésközzel) növelje meg. Ennek értelmében a fenti példa ciklusának magja egymás után ötször fog végrehajtódni, miközben I rendre a következő értékeket veszi fel: Ø, 3, 6, 9, 12

8.Példa

```
vagy{12 FOR X=1 TO 4
vagy{12 FOR X=1 TO 4 STEP 1
    22 PRINT X**X-X;
    32 NEXT X
    42 STOP
```

A programban akár az egyik, akár a másik 12 sorszámú sort alkalmazzuk, a futtatás során kapott eredmények azonosak lesznek. Tehát, ha a ciklusváltozót 1-esével kívánjuk növelni, a STEP alkalmazása szükségtelen.

9.Példa

```
15 FOR C=1 TO 5 STEP 2
25 PRINT C
35 NEXT C
45 STOP
```

Ebben a példában a rendszer a ciklus magját háromszor fogja végrehajtani, miközben C=1,3,5 értékeket veszi fel.

1o.Példa

```
17 FOR T=4 TO -4 STEP -4
21 PRINT T,T**2
25 NEXT T
29 END
```

A 1o. példa azt szemlélteti, hogy a lépésköz értéke negativ is lehet. (Értelemszerűen a kezdő- és végértékként is alkalmazható negativ szám). A példában bemutatott ciklus magja négyszer fog végrehajtódni, miközben T=4,Ø,-4,-8 lesz. A képernyőn megjelenő eredmények a következők lesznek:

4	16
ø	ø
-4	16
-8	64

Végezetül megemlitjük, hogy mind a kezdőérték, mind a végérték, mind pedig a lépésköz megadható kifejezés formájában is. Igy pl. a következő cikluskezdő utasitás is alkalmazható:

```
10 FOR A=I/6 TO I/2 STEP .1*I
```

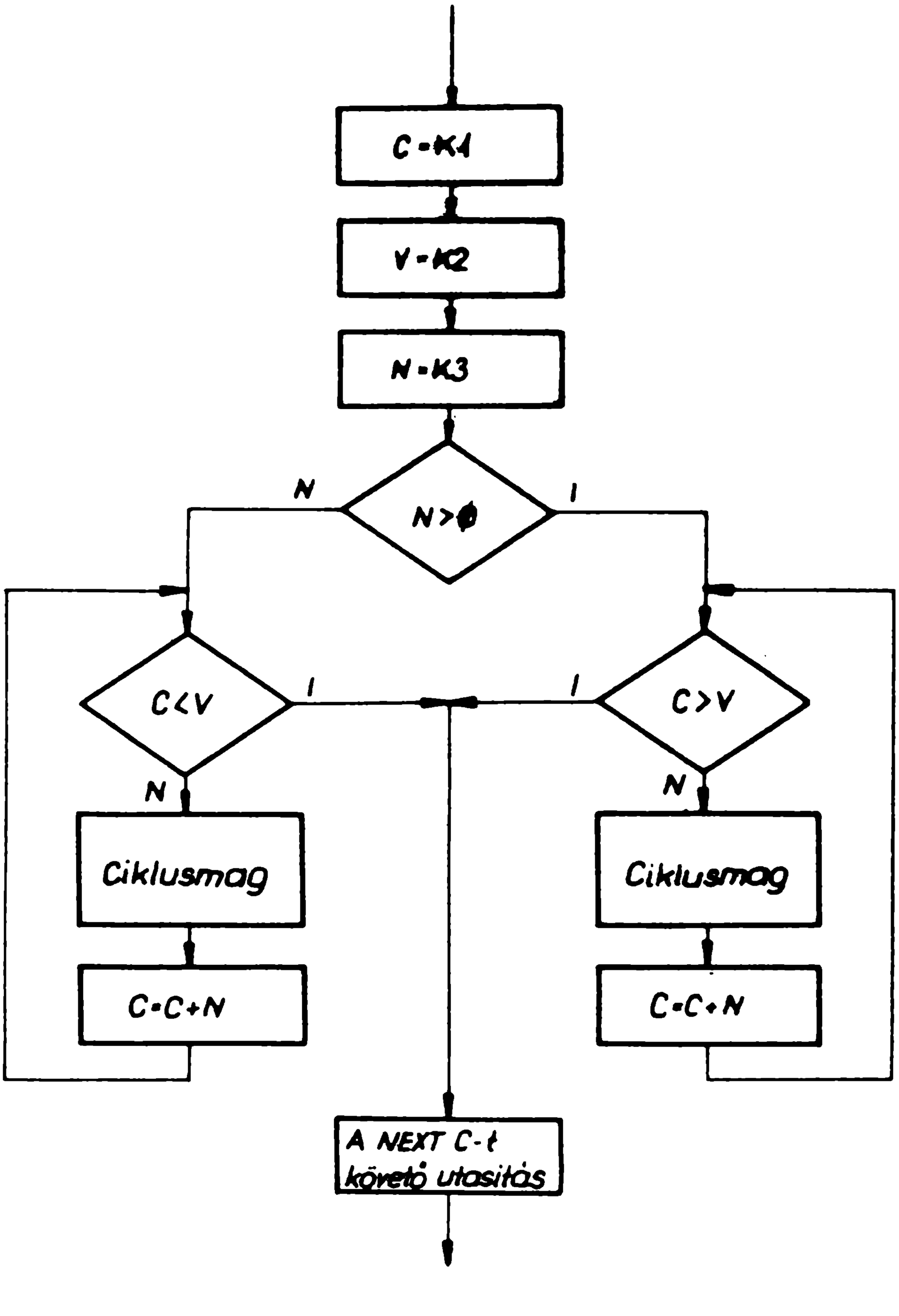
A FOR utasitás legáltalánosabb alakja:

```
cimke  FOR  C=K1  TO K2  STEP K3
```

ahol C: ciklusparaméter (vagy ciklusszámláló, ciklusváltozó)
K1: a ciklusparaméter kezdőértéke (konstans, változó, kifejezés lehet)
K2: a ciklusparaméter végértéke (konstans, változó, kifejezés lehet)
K3: a ciklusparaméter növelési értéke egy ciklus lefutás után (konstans, változó, kifejezés lehet)

Ezeket az általánes jelöléseket felhasználva a FOR-NEXT ciklusutasitások végrehajtási folyamatát a 2.4. ábrán mutatjuk be.

2.4. ábra


C-K1
V-K2
N-K3
N>0
N
I
C<V
I
N
Ciklusmag
C-C+N
I
C>V
N
Ciklusmag
C-C+N
A NEXT C-t
követő utasitás

2.1o. Adatblokk használata - READ,DATA,RESTORE utasitások

A BASIC programokban szereplő konstansok,ill. változók kezdőértékei a program belsejében is elhelyezhetők a DATA utasitás segitségével. A konstansokat, melyek numerikus és szöveges tipusúak is lehetnek, a DATA kulcsszó után egymástól vesszővel elválasztva kell felsorolni.

A DATA utasitás operandusában megadott adatokat nem kell minden egyes programfutásnál INPUT utasitásra begépelni, igy a program belövésének (a helyes működésű program előállitásáig) időszakában sok időt takaritathatunk meg a DATA utasitás alkalmazásával.

A DATA utasitásban felsorolt adatokat a READ utasitással lehet változókhoz hozzárendelni.

Pl.:

```
1ø DATA 2.72,9.81
2ø READ E,G
```

A READ utasitás végrehajtása után az E=2.72, a G=9.81 értékeket vesz fel. A példából látható, hogy a READ végrehajtásakor a rendszer a DATA utasitásban elhelyezett következő elemet hozzárendeli a READ-ben soronkövetkező változóhoz. Ez addig ismétlődik, mig a READ utasitás valamennyi változója értéket nem kap.

A DATA utasitás a programban bárhol elhelyezhető, nem kell hogy megelőzze a benne szereplő konstansokat lekérdező READ utasitást. A gép ugyanis egy előre kijelölt tárterületre helyezi a programban lévő összes DATA utasitásban megadott konstanst természetesen úgy, hogy ennek a tárolóterületnek az első helyére a legkisebb cimkéjű DATA első adata kerül. Ezen az adatmezőn egy mutató halad előre az adatok kiolvasásával együtt

úgy, hogy a mutató mindig az éppen kiolvasandó adatra mutat. Egy változóhoz történő adathozzárendelés után a mutató egy hellyel továbblép, azaz a következő adatot jelöli ki olvasásra. Az adat kiolvasása azonban csak akkor következik be, ha a gépnek ismét READ utasitást kell végrehajtani. Ezek után nyilvánvaló, hogy a READ utasitások listáiban szereplő változókhoz az előfordulásuk sorrendjében rendeli hozzá a gép az adatmező egymás után következő adatait.

A DATA utasitás adattipusai meg kell, hogy egyezzenek a READ utasitás változó listájával; vagyis a READutasitásban a numerikus változó nevéhez numerikus konstanst szöveges változó nevéhez pedig idézőjelek között megadott szöveget kell hozzárendelni.

A DATA és a READ utasitás közvetlen üzemmódban nem alkalmazható.

Ha a DATA utasitás több értéket tartalmaz, mint a program READ utasitása(i), akkor a rendszer a felesleges adatokat nem veszi figyelembe. Ha több változónevet adtunk meg, mint a DATA utasitás(ok) adatainak száma, akkor a rendszer hibajelzést ir ki.

1. Példa

```
10 READ A,B,C
   :
40 READ A1,B1
70 READ F
200 DATA 4,5,6,7,8,9
```

A példában látható, hogy egyetlen DATA utasitáshoz több READ utasitás is tartozhat.

2.Példa

```
12 FOR I=1 TO 5
22 READ XX
32 PRINT XX;' NEGYZETE= ';X**2
42 NEXT I
52 DATA 12%,4%,5%,23%,1
62 END
```

A READ utasitást gyakran használjuk a FOR...NEXT cikluson belül. A ciklus minden egyes végrehajtásakor X-nek új értéke kerül beolvasásra, és egy új sor kinyomtatása történik meg. A program végrehajtásakor az output a következőképpen néz ki:

```
12 NEGYZETE= 143.999
4  NEGYZETE= 16
5  NEGYZETE= 25
23 NEGYZETE= 529
1  NEGYZETE= 1
```

3.Példa

```
7 DATA 3%,4%,8%,16%,7%,24%
17 FOR J=1 TO 3
27 READ X%,Y%
37 PRINT X%,Y%,SQR(X%**2+X%**2)
47 NEXT J
57 END
```

Mivel a DATA utasitás a programban bárhol elhelyezhető, itt a program első utasitása. A programban a FOR... NEXT ciklus háromszor hajtódik végre, és minden egyes alkalommal X és Y új értéket kap. A program végrehajtása után az eredménylista a következőképpen néz ki:

```
3          4          5
8          15         17
7          24         25
```

4. Példa

```
10 READ N
20 FOR P=1 TO N
30 READ D,D1
40 PRINT D**2-D1
50 NEXT P
60 DATA 3
70 DATA 9,1,8,4,7,9
80 END
```

A példában egy READ utasitás van a FOR...NEXT cikluson kivül (ez a végrehajtás során csak egyszer hajtódik végre), a ciklusom belül elhelyezett READ utasitás pedig háromszor (mivel az első DATA utasitás első eleme 3). Az output a következő lesz:

```
80.0008
60.0015
39.9993
```

A RESTORE utasitás lehetővé teszi a DATA utasitások adatainak egy programon belüli többszöri felhasználását. Az utasitás végrehajtása után a következő READ utasitás a legkisebb sorszámú DATA utasitásból olvassa ki az adatokat.
A RESTORE utasitás a következő formákban adható meg:

```
100 RESTORE
```

Ha a 100-as sorszámú utasitást végrehajtja a rendszer, akkor az adatmutató a legkisebb sorszámú DATA utasitás első adatelemére mutat.

Megjegyzés: Figyeljük meg a 2. és a 3. példában a valós és az egész változók alkalmazását!

5.Példa

```
10 FOR I=1 TO 5
20 READ A
30 PRINT A;'NEGYZETE',A**2
40 NEXT I
50 RESTORE
60 PRINT
70 FOR J=1 TO 3
80 READ B
90 PRINT B,'KOBE';B**3
100 NEXT J
110 DATA 4,9,12,8,27
120 END
RUN
4 NEGYZETE= 16
9 NEGYZETE= 81.0008
12 NEGYZETE= 143.999
8 NEGYZETE= 64.0015
27 NEGYZETE= 729.02

4 KOBE= 64.0015
9 KOBE= 729.004
12 KOBE= 1727.99
```

A példában a RESTORE utasitás végrehajtása előtt az adatpointer a 110-es DATA utasitás utolsó adata utánra mutat. A RESTORE utasitás visszaállitja a 110-es sor első adatára a pointert és az adatokat újra kiolvashatjuk.

2.11 Szubrutinszervezés- GOSUB ... RETURN utasitások

A GOSUB utasitás hatására a program normál végrehajtása helyett a GOSUB kulcsszót követő sorszámmal azonositott számú sortól kezdődő programrész -szubrutin- kerül végrehajtásra. A GOSUB-ot követő programsorra a vezér-

lést a szubrutinban elhelyezett RETURN utasitás adja vissza.

A GOSUB és RETURN utasitások szükségtelenné teszik, hogy a programban gyakran alkalmazott részfeladatokat megvalósitó utasitás-csoportokat ismételten beirjuk a programban.

1.Példa

```
10 INPUT A
     :
     :
60 GOSUB 1000
70 PRINT A
     :
200 GOSUB 1000
210 PRINT A,N
     :
990 STOP
1000 N=0
     :
1200 RETURN
1210 END
```

A példában az utasitások végrehajtási sórrendje a következő:

10-60
1000-1200
70-200
1000-1200
210-990

2.példa

```
10 INPUT A,N
        :
        :
60 GOSUB 1000
70 PRINT A,N
        :
900 STOP
1000 IF A*N<150 THEN 1020
1010 GOSUB 2000
1020 A=A+5
        :
1200 RETURN
        :
        :
        :
1990 STOP
2000 A=(A+1)/2
        :
        :
        :
2200 RETURN
2210 END
```

Az utasitások végrehajtási sorrendje az A*N szorzat értékétől függően változhat:

ha A*N < 150	ha A*N ≥ 150
10-60	10-60
1000	1000-1010
1020-1200	2000-2200
70-900	1020-1200
	70-900

Mindkét példában valamennyi szubrutin (az 1.példában 1000-1200-as sorok közti programrész, a 2.példában 1000-1200 és 2000-2200-as sorok közti programrész) előtt egy STOP utasitás helyezkedik el. Igy elkerülhető, hogy a program esetleges hibás végrehajtása esetén 'rácsorogjunk' egy szubrutinra az azt hivó GOSUB utasitás megkerülésével. Ha ugyanis ez az eset előfordulna, a gép egy 29-es hibajelzést - RETURN GOSUB nélkül - irna a képernyőre.

A 2.példában, ha AzN ≥ 15§, szubrutinok egymásba ágyazására - egymásba skatulyázására - láthatunk példát. Itt egy második szubrutin - 2§§§-22§§ közti sorok - kerül végrehajtásra, mielőtt az első szubrutin végét jelző RETURN utasitás hatására kilépnénk az első szubrutinból.

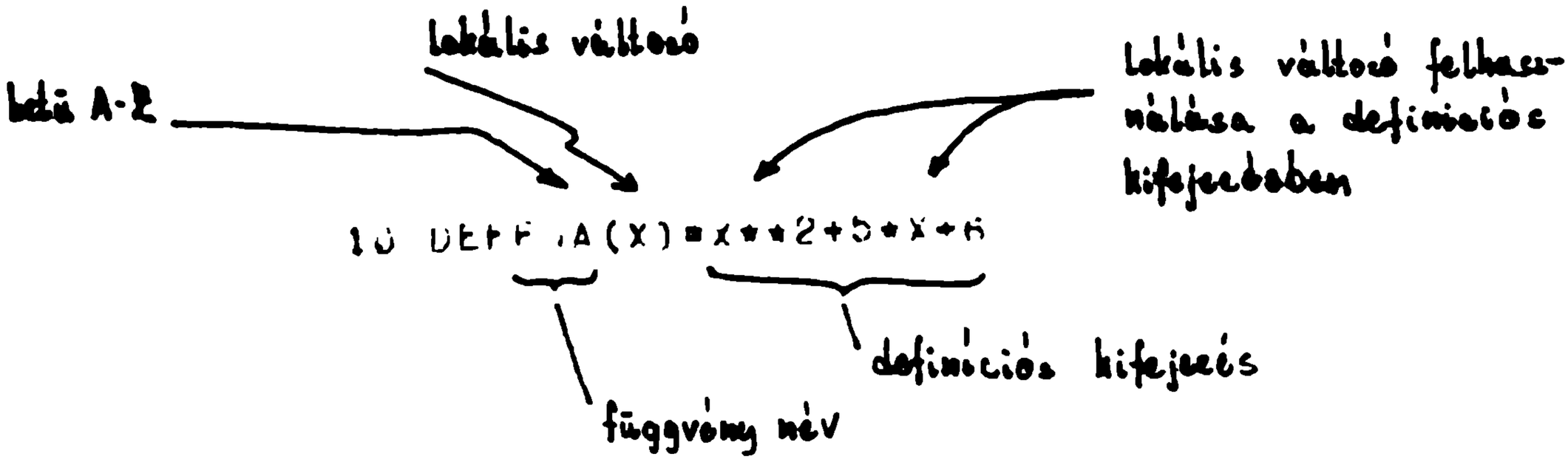
Szubrutinok egynél nagyobb mélységben is egymásba skatulyázhatók. Az egymásba ágyazás mélységét csak a rendelkezésre álló programtároló nagysága korlátozza. (Minden egyes szubrutinhivás 2 további byte-ot foglal le a tárolóban.)

2.12. Függvénydefiniálás - DEF FN utasitás

A DEF FN (Define Function) utasitás segitségével egyszerűsithetjük, áttekinthetőbbé tehetjük a programunkat. Az utasitás segitségével ugyanis egy - a programban gyakran alkalmazott - algebrai kifejezést függvényként definiálhatunk, s a későbbiekben - a definiálás után - a programban már csak a függvény nevét és annak aktuális paramétereit kell leirni a teljes kifejezés helyett.
Aktuális paramétereknek nevezzük a függvény hivásakor felsorolt azon paramétereket, amelyek a függvényben alkalmazott lokális változók helyébe lépnek.

A programban egyidőben maximum 26 függvény definiálható. Ezeket egymástól az angol abc betűivel különböztetjük meg (A-Z). A függvény definiálásakor alkalmazni kell egy - vagy több - egyszerű változót, amelyet lokális változónak nevezünk.

Egy lokális változó csak a függvényen belül definiált. A program egyéb részeiben nincs értéke. Minden más változót globális változónak nevezünk, mert,ha egyszer értéket adtunk neki, azt az értéket a program bármely pontján felhasználhatjuk.



1. Példa

Az 1.példában 'B' névvel definiáltunk egy függvényt, melynek lokális változója az X. A függvény segitségével három értéket számoltattunk és nyomtattunk ki az x=4,5 és 6 aktuális paraméter értékkel. Az x változót, mint lokális változót alkalmaztuk, tehát a függvényen kivül értékkel nem bir. A program futtatása során megjelenő eredmények a következők:

26 30 42

2. Példa

-

A [illegible] egy 'Z' nevu függvényt definiáltunk, Y lok[illegible]zval. (Mint [illegible] látható a függv[illegible] a program [illegible] helyén elhelyezkedés változó, mint [illegible] változó a lokális [illegible] kap értéket [illegible] utasitás által. [illegible] 20 sorszámú sor [illegible] PRINT utasitás kiszámítja a függvény értékét az aktuális paraméter [illegible] vételével, m[illegible] az eredményt meg-

jeleniti. (Ebben és a következő példában az alkalmazott lokális változó, mint globális változó is szerepel a programban.

3.Példa

```
10 INPUT X
20 DEF FNA(X)=SQR(X**2+Y**2)
30 FOR I=1 TO 5
40 INPUT Y
50 PRINT FNA(X)
60 NEXT I
70 END
```

A 3.példában egy 'A' nevű függvényt definiáltunk, melynek lokális változója 'X'. A definiciós kifejezésében egy globális változót -Y- is alkalmaztunk.

2.13. Tömbök - DIM utasitás

Tömböket használunk abban az esetben, ha egy programban a feldolgozás szempontjából szorosan összetartozó adatokkal dolgozunk. Ezeken az adatokon általában ugyanazt a műveletsort kell végrehajtani, de az adatokat tárolni is kell.
Az adatok tárolása változónévvel azonositott tárterületeken történik. Az eddigiek alapján azt tudjuk, hogy egy meghatározott hosszúságú tárrészen (2 vagy 5 byte) egyidőben csak egyetlen érték tárolható, azaz egy változó egy időpillanatban csak egy értéket jelenthet. A feldolgozás egyszerűsitésére lehetőség van különböző tárterületekre is ugyanazzal a névvel hivatkozni, ha megadjuk, hogy azon a nagyobb közös tárterületen belül pontosan hol is helyezkedik el (pl. hanyadik) a feldolgozandó adat. Ezeken a nagyobb, közös tárterületeken elhelyezett adategyütteseket tömböknek nevezzük, az egyes adatokat pedig tömbelemeknek.

Az összetartozó adatok alkothatnak táblázatokat ill. számsorozatokat. Az első csoportba tartozó számtáblázatokat (mátrixokat) kétdimenziós tömböknek, a számsorozatokat (vektorokat) egydimenziós tömböknek nevezzük.

Tömböket programjainkban deklarálás (megadás) után használhatunk. A deklarációs utasitás a DIM, ennek alapján a gép meghatározott tárolóterületet lefoglal a tömb tárolására. A kulcsszó után meg kell adni a tömb nevét - azt ugyanúgy képezzük, mint a változónevet -, majd zárójelben a méretet. A méret egydimenziós tömb esetén egyetlen szám, méghozzá a tömbben előforduló legnagyobb sorszámú elem sorszáma. Kétdimenziós tömb esetén a táblázat maximális sor- és oszlopszámát kell megadni. Egy DIM utasitásban több tömb is meghatározható, ilyenkor az egyes listaelemeket vesszővel kell elválasztani.

Az ABC-80 megengedi azt, hogy a 10-nél kisebb méretű egydimenziós, ill. 10x10-es méretnél kisebb kétdimenziós tömböket ne kelljen deklarálni.

Vigyázni kell arra is, hogy az indexek (sorszámok) 0-val kezdődnek, igy pl. a

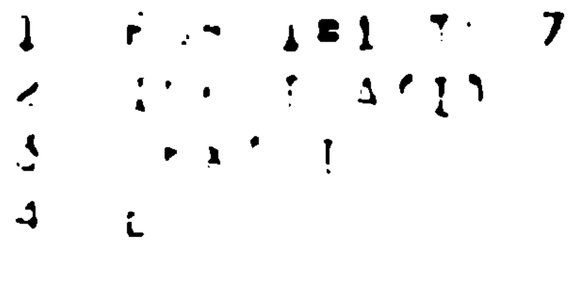
```
20 DIM V(20), M(10,11)
```

utasitásban a V(20) egy egydimenziós, 21 elemű vektornak, az M(10,11) pedig egy 11 sort és 12 oszlopot tartalmazó mátrixnak foglal helyet.

Az adatelemek tömbökbe történő beolvastatását a programba beépitett ciklusokkal végeztethetjük el, legcélszerűbben a FOR...NEXT utasitásokat használva. A következő példákban bemutatunk néhányat az alkalmazási lehetőségek közül.

1. Példa Egydimenziós tömbök (vektorok) kezelése

1.a.



1.b.

Az 1.a. példában a program az adatokat minden egyes ciklus végrehajtásakor a billentyűzetről várja. A Ø 'indexű vektorkomponenshez nem rendelünk értéket.

Az 1.b. példában az adatelemek kiolvasása a programban elhelyezett DATA utasitásból történik.

2. Példa kétdimenzios tömbök (mátrixok) kezelése

A program két FOR ciklus található. A külső, I-re szervezett ciklus a mátrix sorindexét állítja be, a belső J-re szervezett pedig a soron belül vé-

gighaladva az egymásutáni oszlopindexeket képzi, azaz a tömbelemek beolvasása sorfolytonos lesz. I minden egyes értékéhez az J változóra vonatkozó ciklus 6-szor fut le (az A(I,J) tömbnek összesen 24 eleme van).

Tömböket nemcsak numerikus, hanem szöveg tipusú adatok tárolására,azonositására is használhatunk. Szöveget tartalmazó tömbök esetén meg kell adni a szöveges tömb méreteit. Lehetőség van az egyszerű szöveges változó hosszának beállitására is.

3. Példa

```
10 DIM A(16,15),B%(M*N)
```

Az utasitás lefoglal az A kétdimenziós tömb részére 11 x 16=176 helyet, valamint a B, egész számokat tartalmazó vektor részére (m x n + 1) helyet.

4. Példa

```
20 DIM A%=60,C%(40,3)
```

Az utasitás helyet foglal az 1 elemű, maximálisan 60 karakter hosszúságú A szöveges változónak, valamint a 41 x 4 elemet tartalmazó (az egyes elemek max. hossza 80 karakter) C, kétdimenziós szöveges tömb részére.

```
40 DIM D$(30)=6
```

Az utasitás helyet foglal a 31 elemű, egyenként max. 6 karakter hosszúságú D szöveges vektor részére

Megjegyzés: A DIM utasitás a programban tetszőleges helyen elhelyezhető.

Egy programban több DIM utasitás is lehet.

Egy programon belül az egy- és kétdimenziós tömbök változónevei nem egyezhetnek meg. Pl. a következő DIM utasitás használata nem megengedett:

```
10 DIM A(15,15),A(30)
```

2.14. Programfutás megállitása - GET utasitás

A GET utasitás végrehajtásakor a rendszer egyetlen billentyű lenyomására vár, a bevitt karaktert pedig az adott változóhoz rendeli hozzá. A lenyomott billentyű karakterképe nem jelenik meg a képernyőn.

Példa az utasitás alkalmazására:

```
10 GET A%
```

Végrehajtásakor a billentyűzetről bevitt egyetlen karakter hozzárendelése történik meg az A% változóhoz.

Az utasitást általában akkor használjuk, ha a képernyőre kiirandó üzenet (vagy eredmény) nagyobb méretű, mint maga a képernyő. (Pl. 50 soros eredménylista esetén). Ilyenkor számoljuk a kiirt sorokat, és ha ez a szám elér egy előre megállapitott értéket, a vezérlést egy GET utasitásra adjuk. A kiirt sorok kiértékelése után bármelyik billentyűt megnyomjuk, az eredmények kiirása folytatódik.

3. Az ABC 80 rendszer parancsai

3.1. Programok inditása - RUN parancs

A RUN parancs segitségével a tárolóban rögzitett BASIC program végrehajtását indithatjuk meg. Mielőtt a programvégrehajtás megindulna, a rendszer valamennyi változó korábbi értékét törli és a pszeudó véletlenszám generátornak kezdőértéket ad. A program végrehajtása a tárolóban rögzitett BASIC program legalacsonyabb sorszámú sorától indul.

A RUN parancs segitségével lehetőség van egy,a háttértárolón (kazettás adatrögzitő, floppy disk) rögzitett program végrehajtására is. Ez esetben a rendszer a nevezett programot megkeresi a kijelölt háttértárolón, azt a tárba tölti, majd megkezdi a betöltött program végrehajtását. Ha a RUN parancs ez utóbbi formáját kivánjuk alkalmazni, a betöltendő programot, és azt a háttértárolót, amelyen a programot tároljuk, meg kell határozni a rendszer számára. Ilyenkor a RUN parancs alakja a következő:

RUN ' karakterfüzér '

ahol a karakterfüzér a következő elemeket tartalmazza:

- eszköznév (a háttértároló neve)
- file-név (a program neve)
- file-tipus (.BAC vagy .BAS)

(Ld. még 5.1. pontot)

A file-tipus és az eszköznév el is hagyható, ilyenkor a rendszer eszközként a kazettás adattárolót alkalmazza.

pl: RUN CAS: PROGRAM.BAC
RUN PROGRAM
mindkét parancs ugyanazt a programot inditja.
A RUN utasitásként (program sorszámmal) nem alkalmazható.

3.2. Törlés

3.2.1. CLEAR parancs

A CLEAR parancs segitségével törölhetjük az összes változó értékét. A parancs alkalmazása után valamennyi numerikus változó értéke = ø , a karakteres változók értéke üres karaktersorozat lesz.
Utasitásként nem használható.

3.2.2. NEW parancs

A NEW parancs segitségével kitörölhetjük a tárból a tárolt BASIC programot és annak változóit. A parancs végrehajtása során a kazettás adattárolón vagy a floppy diszk egységen megnyitott, de még le nem zárt file-okat a rendszer lezárja.

A parancs hatása egyenértékű az ABC-80 hálózati kapcsolójának ki-, majd bekapcsolásával, illetve a RESET gomb megnyomásával.
Utasitásként nem alkalmazható.

3.3 Programok kilistázása - LIST parancs

A LIST parancs végrehajtásakór a teljes program, vagy annak a specifikált része megjelenik a képernyőn. A parancs a következő formákban adható meg:

LIST

A parancs kilistázza a tárban elhelyezett programot a legkisebb sorszámú utasitássortól a legnagyobbig.

Ha a program sorainak száma nagyobb, mint a képernyőn ábrázolható sorok száma, akkor csak a képernyőn elférő utasitásokat irja ki (az első utasitástól kezdve) a gép. A további programsorok egyenként előhivhatók pl. a szóköz billentyű lenyomásával, ilyenkor a legfelső sorban található utasitás törlődik, mert a képernyő tartalma egy sorral feljebb tolódik. A parancsértelmező módba való visszatérés a RETURN lenyomásával lehetséges.

(Megjegyzés: A program fizikai végét nem az END utasitás jelenti. Ha van még utasitássor az END utasitást követően is, az is megjelenik a képernyőn.)

LIST sorszám

A parancs csak az adott sorszámú utasitássort jeleniti meg a képernyőn.

LIST sorszám, vagy

LIST sorszám -

A parancs hatására a sorszám-mal kezdődő és az utolsó utasitással végződő programrész jelenik meg a képernyőn (ha elfér).

LIST sorszám1, sorszám2 vagy

LIST sorszám1 - sorszám2

A parancs a két sorszámmal meghatározott programrészt listázza ki, beleértve a határokat is.

LIST, sorszám, vagy

LIST- sorszám

A parancs hatására a legkisebb sorszámmal kezdődő és a parancsban megadott sorszámmal végződő programrész jelenik meg a képernyőn. Nem létező sorszámra történő hivatkozáskor a rendszer hibajelzést ir ki. Legyen a tárban a következő program:

```
10 INPUT A
15 IF A=999 THEN 65
20 IF A<0 THEN 50
30 B=A**2-25
40 PRINT 'B= ';B:GOTO 60
50 PRINT 'A SZAM NEGATIV'
60 PRINT 'KEREM A KUVETKEZO ERTEKET':GOTO 10
65 PRINT 'VEGE'
70 END
```

1. Példa

LIST

A parancs végrehajtásakor a teljes program megjelenik a képernyőn:

```
10 INPUT A
15 IF A=999 THEN 65
20 IF A<0 THEN 50
30 B=A**2-25
40 PRINT "B= " ; B :
50 PRINT "A SZAM NEGATIV"
60 PRINT "KEREM A KUVETKEZO ERTEKET" : GOTO 10
65 PRINT "VEGE"
70 END
```

2. Példa

LIST 40

A parancs végrehajtásakor csak a 40-es utasitássorszámú sor jelenik meg a képernyőn:

```
40 PRINT 'B ';B:GOTO 60
```

3. Példa

LIST 50,

A Parancs hatására, a 50-es sorszámú utasitássorral kezdődő, és az utolsó utasitással záruló programrész jelenik meg a képernyőn:

```
50 PRINT 'A SZAM NEGATIV'
60 PRINT 'KEREM A KOVETKEZO ERTEKET':GOTO 1
65 PRINT 'VEGE'
70 END
```

4. Példa

LIST, 40

A parancs végrehajtásakor, a legkisebb sorszámú utasitássorral kezdődő és a 40-es sorral végződő programrész kerül kilistázásra:

```
10 INPUT A
15 IF A=9*9 THEN 65
20 IF A<2 THEN 50
30 B=A**2-25
40 PRINT 'B= ';B:GOTO 60
```

3.4. Programsorok átszámozása - REN parancs

Sok esetben szükség van arra, hogy az utasitások sorszámát megváltoztassuk, hogy további utasitásokat lehessen bevinni a programba. Erre ad lehetőséget a REN (renumber - átszámozás) parancs. A parancs a tárban lévő programot átszámozza valamint átirja a hivatkozási utasitás-sorszámokat is (pl. a GOTO, IF, GOSUB utasitásokban).

A REN parancs a következő formákban adható ki:

REN

A parancs átszámozza a tárban levő teljes programot. Az átszámozott program első sorszáma a 10 lesz, az utasitássorszámok pedig 10 - esével növekednek.

REN sorszám

A parancs átszámozza a tárban levő programot olymódon, hogy a megadott sorszám lesz a program első sorszáma, az utasitássorszámok közötti különbség pedig megegyezik a sorszám értékével.

REN sorszám, lépésköz

A parancs átszámozza a tárban levő teljes programot olymódon, hogy az adott sorszám lesz az új, átszámozott program első utasitássorszáma, az utasitássorszámok közötti különbség pedig megegyezik a lépésközzel.

Legyen a tárban a következő program:

```
2 INPUT A,B,C
3 IF A=0 THEN 20
5 D=A*B-C**2
6 IF D>25.6 THEN PRINT 'AZ ERTEK TUL NAGY!':GOTO 17
11 PRINT 'D=',D
17 PRINT 'KEREM A KOVETKEZO ERTEKEKET!':GOTO 2
20 STOP
```

1. Példa

REN

A parancs végrehajtása után listázáskor a következő program jelenik meg a képernyőn:

```
10 INPUT A,B,C
20 IF A=0 THEN 21
30 D=A*B-C**2
40 IF D>25.6 THEN PRINT 'AZ ERTEK TUL NAGY!':GOTO 17
50 PRINT 'D=',D
60 PRINT 'KEREM A KOVETKEZO ERTEKEKET!':GOTO 2
70 STOP
```

2. Példa

REN. 14

A parancs végrehajtása után listázáskor a következő program jelenik meg a képernyőn:

```
14 INPUT A,B,C
28 IF A=0 THEN 98
42 D=A*B-C**2
56 IF D>25,6 THEN PRINT "AZ ERTEK TUL NAGY" : GOTO 84
70 PRINT "D=",D
84 PRINT "KEREM A KOVETKEZO ERTEKET" : GOTO 14
98 STOP
```

3.Példa

REN 200, 11

```
200 INPUT A,B,C
211 IF A=0 THEN 260
222 D=A*B-C**2
233 IF D>25,6 THEN PRINT "AZ ERTEK TUL NAGY" : GOTO 255
244 PRINT "D=",D
255 PRINT "KEREM A KOVETKEZO ERTEKET" : GOTO 200
260 STOP
```

3.5.Hibakeresés és javitás

3.5.1.Az ED parancs

A parancs lehetőséget ad egy programsor javitására, módositására. A parancs kiadásakor az ED parancsszó után a javitandó programsor sorszámát kell beirni. A RETURN gomb megnyomása után a rendszer kiirja a hivatkozott programsort, majd a cursort a következő sor elejére állitja.

A szerkesztés - javitás során a következőket végezhetjük el:

- cursor mozgatása előre

  Ezt a müveletet a jobbra nyilal (→) billentyüvel hajthatjuk végre. Minden egyes lenyomásra a karakterjelző(cursor) egy karakterrel jobbra elmozdul és a gép kiirja az adott pozición lévő karaktert. (Megjegyzés: a karakterjelző csak

jobbra mozditható el karaktertörlés nélkül!)
Ezekkel a lépésekkel juthatunk el a javitandó programsor első karakterpozíciójától a javitandó rész kezdetéig. A billentyűvel átléptetett karakterek változatlanul megmaradnak a sorban.

- karakter törlése

  Az utasitássorból karaktert a balra nyilat (←) tartalmazó billentyű lenyomásával törölhetünk.

- karakter beszurása klaviaturáról

  A karakterjelzőt arra a pozicióra mozgatjuk el, ahova a karaktereket be akarjuk irni.
  Külön utasitásra nincs szükség, a karakterjelzőtől jobbra elhelyezkedő információ minden egyes billentyűlenyomás után jobbra tolódik és a leütött karakter beiródik.

- a sornak a karakterjelzőtől jobbra levő részének törlése

  A sor egy részének - a karakterjelzőtől jobbra elhelyezkedő információnak - a törlése a RETURN billentyű lenyomásával lehetséges.

  Figyelem: a sor javitásának befejezésekor ügyeljünk arra, hogy ha a teljes sor információjára szükségünk van, akkor a RETURN billentyű lenyomása előtt a karakterjelzőt a sor végére állitsuk be! Ellenkező esetben a karakterjelzőtől jobbra eső információt elveszitjük.

3.5.2. A TRACE parancs

A parancs érvényessége esetén, a program futása közben a végrehajtott utasitások sorszáma megjelenik a képernyőn. Ilymódon a program futásának menete nyomonkövethető.
A parancs utasitásként is kezelhető, ekkor a TRACE szót egy sorszám előzi meg.

Példa a parancs alkalmazására:

```
1[illegible] J=25
2[illegible] K=10
3[illegible] GOTO 7[illegible]
4[illegible] Z=J+K+L+
5[illegible] PRINT "Z=",Z,Z/4
6[illegible] STOP
7[illegible] L=6[illegible]
8[illegible] K=10
9[illegible] GOTO 4[illegible]
10[illegible] END

TRACE
RUN
1[illegible] 2[illegible] 3[illegible] 7[illegible] 8[illegible] 9[illegible] 4[illegible] 5[illegible] Z=          136
34
56
STOP LINE 6[illegible]
```

3.5.3. A NOTRACE parancs

A parancs kiadásakor a program nyomkövetése befejeződik.
A parancs utasitásként is kezelhető, ekkor a NOTRACE szót egy sorszám előzi meg.

Példa a parancs alkalmazására:

```
10  J=25
20  K=15
30  GOTO 70
40  Z=J+K+L+5
50  PRINT 'Z=',Z,Z/4
60  STOP
70  L=60
80  K=10
90  GOTO 40
100 END
45  NOTRACE

RUN
 1   2   3   7.   .     4.   45   70                    134
34
STOP AT LINE 4
```

3.6. Programok mentése és betöltése

3.6.1. A SAVE parancs

A file-ok háttértárolón történő tárolása ill. kinyomtatása a SAVE paranccsal lehetséges.
A parancsban megadjuk az eszköz nevét, a program nevét és a file tipusát. (Ha csak a program nevét adjuk meg, akkor a felvétel automatikusan BASIC file - ként a kazettás egységre történik.)
A parancs operandusai megegyeznek az 5.1. pontbeli operandusokkal.

Példa kazettára történő felvételi paranosra:

```
SAVE PROGRAM
SAVE CAS:PROGRAM.BAC
```

Példa diszkre történő felvételi parancsra:

```
SAVE DRIVPROG
```

3.6.2. A LOAD parancs

A paranos megkeresi a specifikált nevű programot az adott periférián, törli a tárat, majd betölti a programot a tárba. Ügyeljünk arra, hogy kazetta használata esetén a keresés mindig az adott szalagpozicióról kezdődik.
Ha a rendszer megtalálta az adott program elejét, akkor a képernyőn a következő információ jelenik meg:

```
FOUND programnév . file-tipus
```

Ezután kezdődik a program betöltése a tárba. A hangszóróból a töltés ideje alatt az adatátvitel zaját lehet hallani. A program tárba töltése akkor fejeződik be, ha az

```
ABC 80
```

információ megjelenik a képernyőn.

1. Példa

```
LOAD CHECK
```

A parancs végrehajtásakor a mágneskazettán tárolt CHECK nevű program töltődik be a tárba.

2. Példa

```
LOAD CAS:
```

A kazetta adott pozíciója után elhelyezkedő első programot tölti be a tárba a gép.

3. Példa

LOAD DR1 : CHECK

Az 1-es floppy diszken elhelyezett CHECK nevű program töltődik be a tárba a parancs végrehajtásakor.

4. SPECIÁLIS PROGRAMOZÁSI LEHETŐSÉGEK

4.1. A Grafikus üzemmód használata.

Grafikus üzemmódban a képernyőre különböző ábrák, függvények, grafikonok rajzolhatók. A felbontás 72 x 78-as, azaz 72 sort és soronként 78 pontot tartalmaz a képernyő.

4.1.1. Átváltás grafikus üzemmódba.

A rajzolás előtt az adott sort grafikus üzemmódba kell átváltani.
Ez a következő utasitással lehetséges:

```
PRINT CHR$(151)
```

Az utasitás hatására a képernyőnek az a sora, amelyre a PRINT vonatkozott, grafikus módba kerül.
Az egész képernyőterület (24 sor) átváltása pl. a következő ciklussal végezhető el.

```
FOR I=1 TO 24:PRINT CHR$(151);:NEXT I
```

Visszaváltás karakteres módban hasonlóképpen, csak a

```
PRINT CHR$(135)
```

utasitás használatával lehetséges.

4.1.2. Rajzolás a képernyőre a grafikus utasitásokkal.

A rajzolásra két utasitás és egy függvény szolgál.

SET DOT sor, oszlop	-	egy pont megjelenitése, kivilágitása a képernyőn
CLR DOT sor, oszlop	-	egy pont törlése, elsötétitése a képernyőről
DOT (sor, oszlop)	-	egy a képerőny lévő,

pont állapotának lekérdezése

a függvény értéke IGAZ, ha a pont világit és HAMIS ha a pont nem világit.

sor = Ø - 71 közti értékek

oszlop = 2 - 79 közti értékek

Minden sor Ø. és 1. pozíciója a gép számára van fenntartva, rajzolásra nem használható.

Az előző megadásból következik, hogy például a

SETDOT Ø, 2 utasitás hatására a bal felső sarokban lévő pont fog világitani

A következő utasitás sor ugyanezen pont állapotát fogja megváltoztatni.

[illegible]

Az alábbi mintapélda a grafikus üzemmód használatát mutatja be. Inditás után az egész képernyőt grafikus módba váltja át, majd egy spirálisan kifelé haladó ábrát állit elő.

[illegible]

4.1.3. Rajzolás a képernyőre fél-grafikus karakterekkel.

A grafikus módba történő átváltás után a PRINT

parancs segitségével is tudunk a képernyőre rajzolni, ekkor azonban csak 24 sor és 39 karakterpozició áll rendelkezésünkre.
Amint az a 4.1. táblázatban látható, az egyes kódokhoz más szimbólum tartozik karakteres és más grafikus módban. Az itt látható félgrafikus szimbólumok viszont hatszor akkorák mint az előbb emlitett SETDOT és CLRDOT utasitással kezelhető grafikus pontok. Megfelelő kombinációjukból különböző ábrák épithetők fel.

A következő programsorral egy vékony vizszintes vonal huzható a képernyőn a cursor poziciójától kezdődően. Ezt a 112-es kód többszöri kinyomtatásával érjük el.

Ha ugyanebben a programsorban a 124-es kódot használjuk, akkor a vonal vastagabb lesz.

Amint a táblázatból kitünik, a nagybetüknek nincs grafikus megfelelőjük, tehát a különböző ábrákba feliratokat is elhelyezhetünk.

ASCII kód	Karakter	Grafikus kép.	ASCII kód	Karakter	Grafikus kép.	ASCII kód	Karakter	Grafikus kép.	ASCII kód	Karakter	Grafikus kép.
32	Blank		56	8		80	P	P	104	h	
33	!		57	9		81	Q	Q	105	i	
34	"		58	:		82	R	R	106	j	
35	#		59	;		83	S	S	107	k	
36	¤		60	<		84	T	T	108	l	
37	%		61	=		85	U	U	109	m	
38	&		62	>		86	V	V	110	n	
39	'		63	?		87	W	W	111	o	
40	(		64	É	É	88	X	X	112	p	
41	)		65	A	A	89	Y	Y	113	q	
42	*		66	B	B	90	Z	Z	114	r	
43	+		67	C	C	91	Ä	Ä	115	s	
44	,		68	D	D	92	Ö	Ö	116	t	
45	-		69	E	E	93	Å	Å	117	u	
46	.		70	F	F	94	Ü	Ü	118	v	
47	/		71	G	G	95	–	–	119	w	
48	0		72	H	H	96	é		120	x	
49	1		73	I	I	97	a		121	y	
50	2		74	J	J	98	b		122	z	
51	3		75	K	K	99	c		123	ä	
52	4		76	L	L	100	d		124	ö	
53	5		77	M	M	101	e		125	å	
54	6		78	N	N	102	f		126	ü	
55	7		79	O	O	103	g		127	■	

Karakteres kódok ASCII és grafikus megfelelõi.

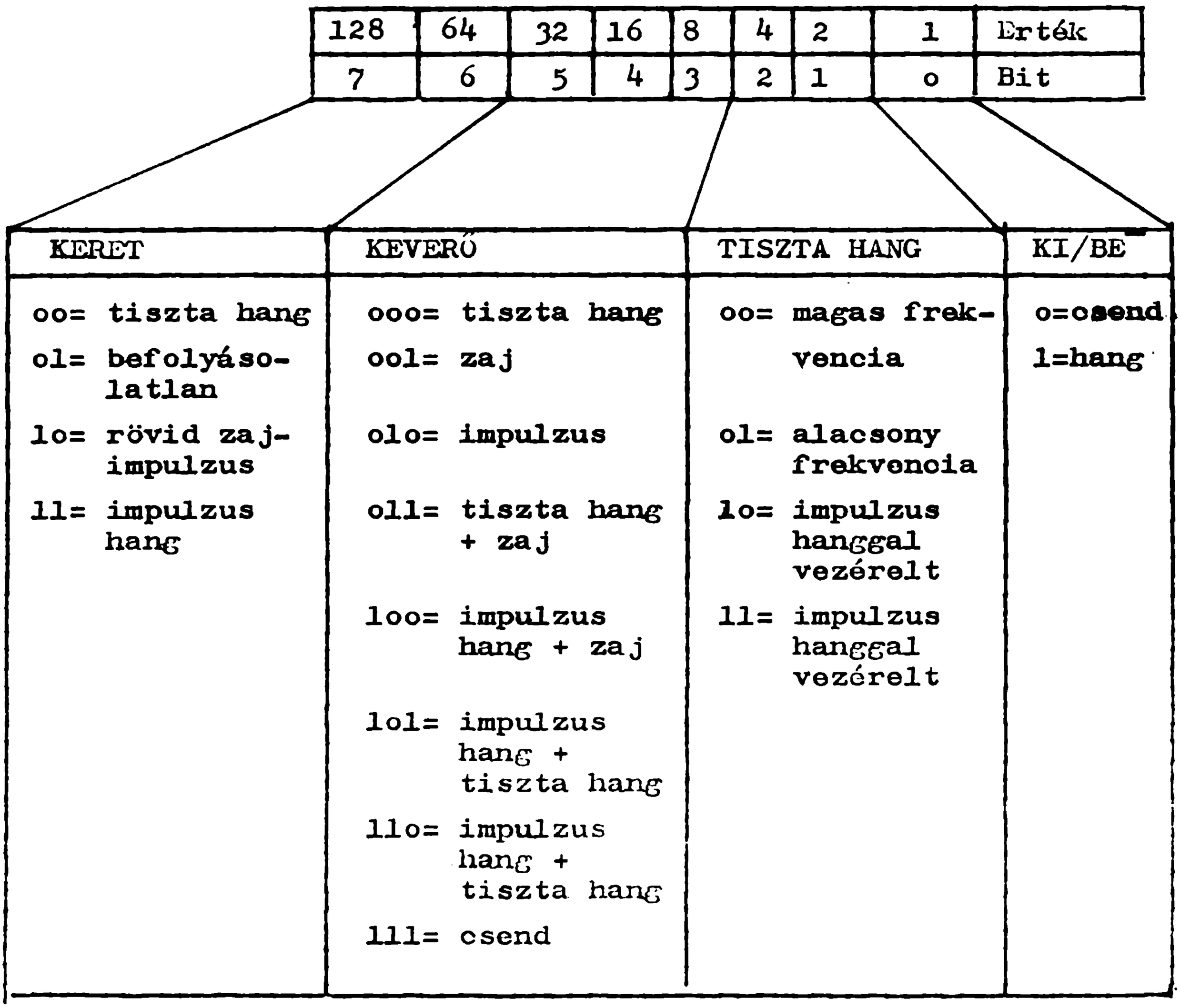
4.1. táblázat.

4.2. Hangefektusok előállitása.

A programozható "hangefektus-generátor" három fő részből áll.

- tiszta hang generátor
- zajgenerátor
- keverő

A generátor az OUT 6, adat paranccsal vezérelhető, ahol az "adat" értéke határozza meg az előállitandó hangefektust.
Ezen adat értéke az adott hatás kiválasztásával a 4.2. táblázatból határozható meg.

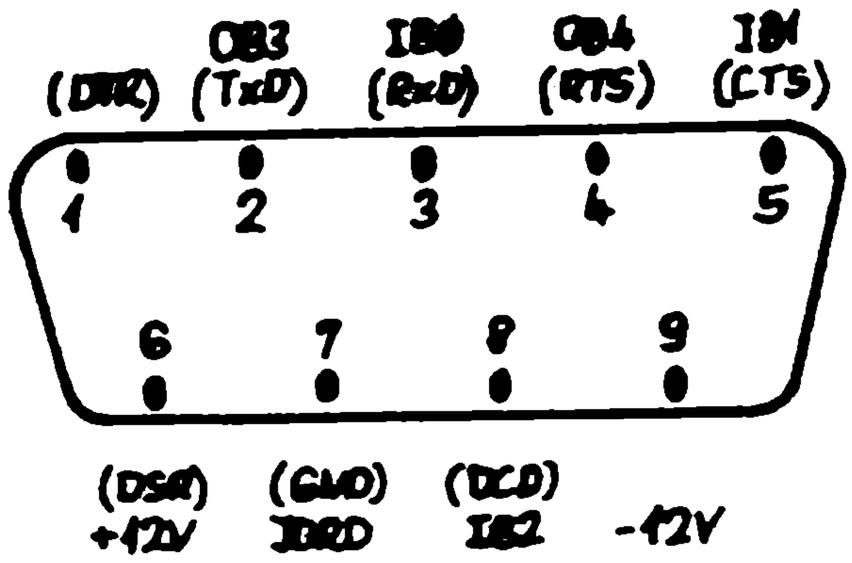


4.2. táblázat.

Pl. egy sziréna hangját az OUT 6,7 paranccsal állithatjuk elő.
Egy bekapcsolt hangefektus mindaddig szól, mig azt az OUT 6,Ø paranccsal ki nem kapcsoljuk vagy más OUT paranccsal nem határozunk meg ujabb hangot.
Az OUT utasitásként is használható, azaz a hangefektusok keltése és megszüntetése programokban is lehetséges.

4.3. A V24 -soros interface programozása.

A gép hátoldalán található egy soros-interface csatlakozó, melynek felhasználásával az ABC-8o-hoz egyéb berendezések csatlakoztathatók.
Ezen csatlakozón keresztül relék, lámpák stb. is vezérelhetők. A csatlakozó bekötését a 4.1. ábrán mutatjuk be:



4.1. ábra.

A csatlakoztatásnál ügyeljünk arra, hogy a gép kimeneti jelszintje ± 7V, ami max 1o mA-vel terhelhető!

A bemeneti jelszint ± 2,5 V-tól- ± 3o V-ig terjedhet, ahol a negativ feszültség jelenti a logikai 1 -et.

A soros interface az 58-as periféria cimre van bekötve, tehát az OUT 58, adat paranccsal vezérelhető, vagy az INP (58) függvénnyel kérdezhető le.

A kazettás adattároló motorjának bekapcsolása például az OUT 58, 32 paranccsal érhető el.

Az INP és OUT parancsokban ill. utasitásokban használatos adat-funkció kapcsolatot a 4.3. táblázatban adjuk meg.

4.3. Táblázat:

Megnevezés	bit	érték
szalag be	7	128
szalag ki	6	64
Motor Relé	5	32
RTS ki	4	16
TXD ki	3	8
DCD be	2	4
CTS be	1	2
RXD be	0	1

4.4. A kettős kazettás adattároló kezelése.

Az ABC 8o a hozzácsatolt kettős kazettás adattárolóból egyszerre mindig csak az egyik egységgel képes kapcsolatot tartani.
Az egységek közötti átkapcsolást OUT utasitásokkal lehet végrehajtani.

A hálózat bekapcsolása után mindig az 1-es egység lesz üzemképes állapotban.

A 2-es egység kijelölése:

```
OUT  58,  16
OUT  58,  24
```

Az 1-es egység kijelölése:

```
OUT  58,   Ø
OUT  58,   8
```

A klaviatura hátoldalán található RESET gomb megnyomására az 1 -es kazettás egység kapcsolódik a központi egységhez.

4.5. A gépi szintü programozást támogató utasitások.

A következő utasitások és függvények lehetőséget adnak kisebb gépi szintü programok memóriába irására és azok aktivizálására a BASIC programból. A szögletes zárójelbe irt elemeknek nem kell kötelező érvénnyel szerepelni az utasitásokban.

POKE tárcim, adat1 [,adat2 ....]

Az utasitás hatására az adott tárcimtől kezdődően eltárolja az adat1 , adat22 stb. értékeket.

Az adat értéke Ø és 255 közé eshet.

PEEK (tárcim)

a függvény értéke a tárcimmel adott rekesz tartalmával lesz egyenlő.

OUT periféria szám, adat1

Az utasitás hatására az adott perifériára kiküldi adat1 értékét.

Az adat értéke Ø és 255 közé eshet.

INP (periféria szám)

A függvény értéke az adott perifóriáról érkező adat értékével lesz egyenlő.

SWAP (adat)

A függvény értéke az operandusként megadott szám 2 byte-os egész értékével lesz egyenlő, de a magasabb és alacsonyabb helyiértékü byte-ok felcserélődnek.

CALL (tárcim)

A függvény aktivizálja az adott tárcimen kezdődő gépi szintü programot, majd a BASIC programba visszatérve a mikroprocesszor HL regiszterpárjában lévő adat értékét veszi fel.

CALL (tárcim, adat)

Az előző függvénnyel azonosan müködik, csak a gépi szintü program aktivizálása előtt az adat értékét a mikroprocesszor DE regiszterpárjába tölti be.

Az adat értéke 2 byte-os egész lehet.

4.6. ASCII ARITMETIKA

Az 1. pontban részletesen bemutatott számábrázolá-
son és a hozzá kapcsolódó aritmetikán kivül az
ABC-80 BASIC lehetőséget ad max. 29 számjegyet tartalma-
zó numerikus mennyiségekkel végzett alapmüveletek
eredményeinek előállitására is.

Ebben az esetben a számok tulajdonképpen speciális
szövegkonstansok, amelyek csak számjegyeket illetve
előjelet és tizedespontot tartalmazhatnak.
Exponens (kitevőrész) megadására nincs lehetőség.

A négy alapmüvelet ezen számok között függvények
segitségével hajtható végre.

Ezen függvények a következők:

ADD$ - összeadás
SUB$ - kivonás
MUL$ - szorzás
DIV$ - osztás

Az operandusmező (argumentum) mindegyik függvény-
nél azonos felépitésü, a két szöveges változó után
meg kell adni egy 1-től 29-ig terjedő számot, ami
azt mutatja, hogy az eredmény hány számjegyre kere-
kitve keletkezzék. Általánosan felirva pl.

A$=ADD$(B$,C$,N)

A következő programrész az ASCII aritmetikára mutat példát:

```
90 B$='123456789012'
100 C$='987654321'
110 A$=ADD(B$,C$,8%)
120 PRINT 'A$=';A$
```

A$=12355555

A négy alapműveleten kivül az ASCII aritmetika lehetőséget biztosit két karakterfüzér számként történő összehasonlitására is.

K%=COMP%(A$,B$)

Itt k% értéke az operandusoktól függően a következő lehet:

k% = -1 % ha A$ < B$

k% = 0 % ha A$ = B$

k% = 1 % ha A$ > B$

A két karakterfüzér összehasonlitása során először az első karakterek ASCII kódjait hasonlitja össze és a kódok értéke alapján eldönti, hogy melyik karakterfüzér a kisebb.

Amennyiben ezen karakterek kódjai egyformák, ugy a következő egy-egy karaktert vizsgálja meg. Ezt végzi mindaddig, amig vagy elfogy az egyik karakter füzér, vagy különböznek egymástól a karakterek.

Azonos karakter füzérek esetén a rövidebb a kisebb.

<u>Pl:</u> a következő logikai kifejezések igazak:

" ABCD" < " ZQY."

" AKLM" = " AKLM"

" AKLM" > " AKL "

4.7. Logikai kifejezések

Az 1. fejezetben bemutatott konstansok és változók mellett az ABC-80 BASIC logikai konstansok és változók kezelésére is képes. Logikai adatnak egy bináris kétértékü jelet tekinthetünk ugy, hogy a jel egyik megjelenési formájához (pl. a Ø számjegyhez) a HAMIS (FALSE) logikai értéket, mig másik megjelenési formájához (pl. az 1 számjegyhez) az IGAZ (TRUE) logikai értéket rendeljük. Egy logikai adat tehát egyetlen biten ábrázolható. Az ABC-80 BASIC-ben a legkisebb, együtt kezelhető adathossz 16 bit, igy gép számára egy logikai konstans is 16 bites lesz ugyanugy, mint egy egész tipusu konstans.

A logikai konstansok értelmezési tartománya tehát az

ØØØØØØØØØØØØØØØØ - 1111111111111111

intervallum, értékkészletükbe viszont csak két érték, az IGAZ ill. HAMIS.

A HAMIS állapot megvalósitására egyetlen egy 16 bites szám, a 16 db zérust tartalmazó bitsorozat szolgál, még az összes olyan bitsorozat, amely legalább egy egyest tartalmaz, a logikai IGAZ értéket képviseli.

Logikai konstansokat BASIC programjainkban automatikusan is kaphatunk, hiszen a relációk kiértékelésekor annak igaz, vagy hamis voltától függően vagy IGAZ vagy HAMIS értékü adattal találkozunk. Az IGAZ érték 16 db egyest, a HAMIS 16 db Ø-t tartalmaz.

Logikai konstansok, változók között logikai müveleteket értelmezhetünk. Az ABC 8o BASIC a következő logikai müveletek végzésére ad lehetőséget.

Müveletet jelző kulcsszó	Operandus-szám	A müvelet értelmezése
AND	2	A müvelet eredménye IGAZ, ha mindkét operandus értéke IGAZ
OR	2	IGAZ, ha legalább az egyik operandus IGAZ
XOR	2	IGAZ, ha csak az egyik operandus IGAZ
IMP	2	HAMIS, ha az elsőnek irt operandus IGAZ és másodikként szereplő hamis, minden más esetben IGAZ.
EQV	2	IGAZ, ha a két operandus egyenlő értékü
NOT	1	IGAZ, ha az operandus HAMIS, HAMIS ha az operandus IGAZ

Logikai kifejezéseket logikai konstansok, változók és müveletek használatával állithatunk elő.

Pl. ha a vezérlést akkor akarjuk átadni a 1ØØ-as utasitásra, ha az A változó értéke negativ, nulla vagy ötnél nagyobb, akkor a következő utasitást érdemes felirnunk:

```
3Ø  IF  A<=Ø  OR  A>5  THEN  1ØØ
```

5. FILE KEZELÉS

Az ABC-80-hoz csatlakoztatható háttértárolókon (kazettás magnó és floppy-diszk) adatok és programok egyaránt tárolhatók.
Az adatok tárolása file-okban történik.
Mielőtt egy file-ba irunk vagy abból olvasunk, a file-t meg kell nyitni és a tényleges átvitel (irás vagy olvasás) teljes befejezése után le kell zárni.

Ezen feladatok ellátására a következő utasitások szolgálnak:

5.1. File nyitás és zárás.

OPEN karakterfűzér AS FILE file-szám

Az utasitás hatására a gép karakterfüzérben megadott nevű file-t olvasásra megnyitja és az adott file-számot hozzárendeli.

A karakterfüzér tartalmazza:

- az eszköznevet, amely a következő lehet

DR Ø :
.
.
.
DR 6 :

floppy diszk esetén

CAS : - kazetta esetén

PR : - nyomtató esetén

- a file nevet, amely maximálisan 8 karakterből állhat. A nevet kötelezően betűvel kell kezdeni, a többi pozición számok is szerepelhetnek. Az ékezetes betűk használata nem megengedett

- a file tipust, amely három karakterből állhat.

Amennyiben a file tipusát külön nem jelöljük, ugy az alábbi tipusokat rendeli a gép automatikusan a file-hoz.

SAVE utasitásnál .BAC - BASIC (átmeneti kódu)

LIST utasitásnál .BAS - BASIC (forrás kódu)

LOAD utasitásnál .BAC vagy .BAS

UNSAVE utasitásnál .BAC vagy .BAS

A file-szám értéke 1 - 255-ig terjedhet.

Az OPEN utasitás végrehajtása során tényleges átvitel (I/O művelet) nem történik, csak a megadott paraméterü file-hoz hozzárendeli a rendszer az utasitásban szereplő file-számot.

Irás vagy olvasás esetén a későbbiekben erre a file-ra mindig ezzel a file-számmal kell hivatkozni.

PREPARE karakterfüzér AS FILE file-szám.

Az utasitás hatására a karakter-füzérben megadott nevü file-t irásra megnyitja és az adott file-számot rendeli hozzá.

Az operandusmező megadását lásd az OPEN utasitásnál.

CLOSE file-szám

Ezen utasitás segitségével zárhatók le az irásra vagy olvasásra megnyitott file-ok.

Minden file-t, melyet irásra megnyitottunk a kiirás befejeztével le kell zárni, különben a kiirt tartalom elvész.

Példák az előző utasitásokra.

```
10 OPEN 'DRW:ADAT.TXT' AS FILE 1
```

A 1Ø-s cimkéjü BASIC sorral a nullás diszk egységen olvasásra megnyitjuk az ADAT nevü TXT tipusú file-t, melyre a későbbiekben az egyes file-számmal hivatkozhatunk. A file tipusa a file jellegének jelölésére szolgál, elsősorban a felhasználó tájékozódását segiti elő.

```
20 PREPARE 'CAS:PROBA.BAS' AS FILE 255
```

Ebben a példában irásra nyitottunk meg egy file-t a kazettás magnón PROBA névvel. A tényleges irás müveletnél elég csak a fileszámmal (255) hivatkozni a file-ra.
(Ld. 5.2.pont)

```
[illegible] CLOSE 255
```

A file lezárásakor a file nevét nem kell megadni, csak azt a file-számot, amit a megnyitáskor hozzárendeltünk.

5.2. Irás illetve olvasás a file-ból

INPUT ≠ file-szám, numerikus vagy karakteres változók.

Az INPUT utasitással adatokat nem csak a klaviaturáról olvashatunk be, hanem valamelyik, általunk olvasásra megnyitott és nem üres file-ból is.

Az utasitás végrehajtása során a felsorolt változók, melyeket egymástól vesszővel kell elválasztani, értéket kapnak.

Például:

```
10 INPUT#1,A1,X,Q$,W(1,5)
```

Ha az 1 file-számu file az INPUT ≠ 1 ... utasitás kiadásakor a 239, 2.72E5, "KEZDET",-19.5 ... adatokat tartalmazta, akkor az utasitás végrehajtása az A1=239,X=2.72E5, Q$="KEZDET", W(1,5)=-19.5 hozzárendeléseket jelenti.

INPUT LINE ≠ file-szám, karakteres változó.

A fenti utasitás a megadott számu file-ból egy karakteres változóhoz rendel értéket.

Az INPUT utasitástól eltérően a hozzárendelt érték a vezérlőjeleket (kocsi-vissza, soremelés) is tartalmazza.

Például:

```
10 INPUTLINE#255,K$
```

PRINT ≠ file-szám, numerikus vagy karakteres változók ill. konstansok.

Az utasitás hatására a megadott file-ba kii-

rásra kerülnek az operandusmezőben felsorolt változó- ill. konstans értékek. Természetesen az irás megkezdése előtt a file-t egy PREPARE utasitással meg kell nyitni.

Például:

```
PRINT #2,'TELJES ÖSSZEG:';S%
```

Az utasitás végrehajtásának eredményeként a 2 file-számu file-ra kiiródott a "TELJES ÖSSZEG" és az 5 % változó értéke egy rekordba.

5.3. Programok rögzitésére és betöltésére szolgáló parancsok

LIST karakterfüzér

A parancs, ha a parancsszó után file név áll a memóriában lévő programot nem a képernyőre, hanem a megadott eszközre, a megadott file-ba veszi fel (listázza vissza). A LIST és a SAVE parancs között az a különbség, hogy mig a SAVE a tárban lévő BASIC programot ugynevezett leforditott, azaz átmeneti kódban lévő formájában tárolja, addig a LIST parancs ezt az átmeneti kód formát előbb visszaforditja forrás szöveggé (ASCII kódba) és ezt rögziti.

Ezen parancs segitségével listázható ki a program a nyomtatóra, ha eszköznévként PR: - t adunk meg. Ilyenkor a filenév és file tipus elmaradhat.

A karakterfüzér megadásának szabályát lásd az OPEN-nél.

MERGE karakterfüzér.

A parancs segitségével két programot tudunk a memóriában összeszerkeszteni. Ugyanis a MERGE parancs a program betöltése előtt nem törli a memóriát s igy a betöltendő program a már bent-lévőhöz hozzámásolódik.
Amennyiben a két programban azonos sorszámu sorok vannak, ugy a betöltendő program adott sora felülirja a tárbanlévő azonos sorszámu sort. A karakterfüzér megadásának szabályait az OPEN utasitásnál mutattuk be.

CHAIN karakterfüzér

Ezen utasitás lehetőséget ad arra, hogy a tul hosszu programokat több részprogramra bontsuk és azokat egymás után lefuttassuk. A CHAIN utasitás, tehát programban is kiadható, ezért ez a folyamat teljesen automatikusan is lejátszódhat, vagyis az előző programrész legvégén (a legutolsó utasitásban) megadhatjuk, hogy hol helyezkedik el az a programrész, amellyel a programnak folytatódnia kell. Először ez a programrész betöltődik a memóriába, majd rögtön el is indul. Ügyeljünk arra, hogy a CHAIN utasitás a végrehajtás első lépéseként a régi programot és az ahhoz tartozó adatokat kitörli, igy az adatok átadása csak nehezen (POKE és PEEK utasitások segitségével) valósitható meg.
A karakterfüzér értelmezését ld. OPEN utasitásnál.

Példa két programrész láncolására.

Tételezzük fel, hogy egy MICRO nevü programot két részre kellett osztanunk: MICRO1 és MICRO2-re, melyeknek egymás után kell lefutniuk.
A MICRO1 programrésznek a következő adatokat kell átadni a MICRO2 számára:
A1, B1, Ö%, S(1), S(2), ... S(1o).
Az első programrész logikai végére a következő adattároló programrészt kell irni az 1øøø-es cimkétől kezdődően.

```
10 REM *** MICRO 1 ***

1000 REM MICRO 1 PROGRAM VEGE ,ADATOK TAROLASA
1010 E%=65408%:REM A PIKE TERULET KEZDOCIME
1020 T=A1:GOSUB 3000:REM A1 TAROLASA
1030 T=B1:GOSUB 3000:REM B1 TAROLOSA
1040 T%=0%:GOSUB 5000:REM 0% TAROLASA
1050 FOR I%=1% TO 10%
1060 T=S(I%):GOSUB 3000:REM S(I%) TAROLASA
1070 NEXT I%
1080 CHAIN 'MICRO2':REM A MASODIK RESZ BETOLTESE ES INDITASA

3000 REM * VALOS SZAM TAROLASA *

5000 REM * EGESZ SZAM TAROLASA *
```

A MICRO2 programrész elején az adatok beolvasására a következő rutinnak kell állnia:

```
10 REM *** MICRO 2 ***
20 REM AZ ELTAROLT ADATOK VISSZATOLTESE
30 E%=65408:REM AZ ELSO ADAT CIME
40 GOSUB 4000 : A1=T:REM A1 TOLTESE
50 GOSUB 4000 : B1=T: REM B1 TOLTESE
60 GOSUB 6000 :0%=T: REM 0% TOLTESE
70 FOR I%=1% TO 10%
80 GOSUB 4000 : S(I%)=T : REM S(I%) TOLTESE
90 NEXT I%
100 REM * A MICRO 2 PROGRAM KEZDETE *

4000 REM

5000 REM
```

Az adattároló illetve adatbeolvasó programrészekben használt szubrutinok:

```
5000 REM * EGESZSZAM TAROLASA *
5010 POKE E%,SWAP%(T%),T%
5020 E%=E%+2%
5030 RETURN
```

```
6000 REM * EGESZSZAM TOLTESE *
6010 T%=SWAP%(PEEK(E%))+PEEK(E%+1%)
6020 E%=E%+2%
6030 RETURN
```

```
3000 REM * VALUSSZAM TAROLASA *
3010 I$=NUM$(T)
3020 FOR I%=0% TO LEN(I$) -1%
3030 POKE E%+I%,ASC(MID$(I$,I%+1%,1%))
3040 NEXT I%
3050 POKE E%+I%,255%
3060 E%=E%+I%+1%
3070 RETURN
```

```
4000 REM * VALOSSZAM TOLTESE *
4010 J$=' '
4020 FOR U%=E% TO 65535%
4030 IF PEEK(U%)=255% THEN 4060
4040 J$=J$+CHR$(PEEK(U%))
4050 NEXT U%
4060 T=VAL(J$):E%=E%+LEN(J$)+1%
4070 RETURN
```

5.4. Speciálisan diszk egységre értelmezett parancsok.

A következő utasitások kizárólag diszkes egységekre adhatók ki (DRØ - DR6)

UNSAVE karakterfüzér

A parancs segitségével adott file törölhető ki a diszkről. A file tipus megadása nem kötelező, ekkor .BAC vagy .BAS tipusu file-t fog törölni a gép.°

KILL karakterfüzér

Hatása ugyanaz, mint az UNSAVE parancsé, de a file tipus megadása mindig kötelező.

A KILL utasitásként is megadható.

NAME karakterfüzér AS karakterfüzér

A diszken az adott file nevét kicseréli az AS szócska után álló névre. A NAME parancsként és utasitásként is kiadható.

Figyelem: a két eszköznek azonosnak kell lennie!

6. Néhány programszerkesztési recept.

6.1. Adatbevitel.

Megoldandó feladataink többségében szükség van adatbevitelre, ennek a munkafázisnak jó szervezése elengedhetetlen programjainkban. Attól függően, hogy több azonos tipusu és a feldolgozás szempontjából egységesen kezelendő adatról vagy egymással formailag és tartalmilag nem szorosan kapcsolódó adatokról van szó, ciklikus ill. nem ciklikus(egyedi) adatbeviteli programrészt állithatunk elő.

6.1.1. Egyedi adatbevitel.

Az adatbevitelt általában INPUT vagy READ-DATA utasitásokkal oldjuk meg.
Ha INPUT utasitást használunk, akkor az adat begépelését a felhasználótól várjuk. Annak érdekében, hogy programunkat ne csak saját magunk, hanem bárki használni tudja, az INPUT utasitásra beirandó adat valamilyen specifikációját a felhasználóval közölni kell, erre nyilvánvalóan egy szövegkonstanst (mint üzenetet) kiiró PRINT utasitást célszerü használni. Ebben a tájékoztató szövegben a lehető legpontosabban, legegyértelmübben (az észszerüség határain belül!) próbáljuk meg informálni a felhasználót a tőle várt adatmegadásról.

Az adat begépelését követően vizsgáljuk meg azt, hogy az adat valóban megfelel-e bizonyos alapvető, vele szemben támasztott követelményeknek.
Alapvetőnek minősithető pl. az, hogy a beirt adat tipusa (egész, valós, szöveges) megfelel-e a várt tipusnak. Ha ez a megfelelés nincs meg, a gép automatikusan hibaüzenetet ad. Ha az ilyen tipusú

hibára külön üzenettel is fel akarjuk hivni a felhasználó figyelmét, az ON ERROR GOTO utasitást használhatjuk.

A gép által automatikusan nem felismerhető hibákra külön hibafelismerő programrészt kell irnunk. A vizsgálatot leggyakrabban intervallumra vagy konkrét értékre (pl. egy szöveges adat értéke "IGEN"-e) végezzük. A vizsgálat bonyolultságától függően több IF utasitást tartalmazó utasitássor megirására is sor kerülhet.

A következőkben olyan programrészt mutatunk be, amely alkalmas arra, hogy segitségével egy gépkocsi sebességének értékét vigyük be m/s egységet használva.

```
100 **A GEPKOCSI SEBESSEGENEK BEOLVASASA **
110 PRINT "IRJA BE A GEPKOCSI SEBESSEGET M/S - BAN !"
120 INPUT V
130 ON ERROR GOTO 170
140 IF V<0 THEN 190
150 IF V>40 THEN 210
160 GOTO 230
170 PRINT "A SEBESSEG NUMERIKUS ERTEK! SZAMOT KEREK"
180 GOTO 120
190 PRINT "A SEBESSEG NEM LEHET NEGATIV! UJ ADATOT KEREK!"
200 GOTO 120
210 PRINT "A KOCSI SEBESSEGE TUL NAGY (>144 KM/ORA)! UJ ADATOT!"
220 GOTO 120
230 REM ** A SEBESSEG BEVITEL VEGE **
```

6.1.2. Összetartozó adatok bevitele.

Az összetartozó adatok bevitele tulajdonképpen egyedi adatok bevitelének automatikus ismétlése, ezért itt is érvényesek a 6.1.1. pontban leirtak.

Azokon tul azonban kiilön is érdemes figyelni az ilyen adatbevitelt, ugyanis az ismételt adatbeirásoknál előfordulhat az az eset, hogy ismerjük a beviendő adatok számát, de az sem kizárt, hogy a begépelendő adatok számát előre nem ismerjük. Nézzük meg, hogyan kell eljárni ebben a két különböző esetben.

Ha a beolvasandó, azonos tipusu, formáju, értéktartományu adatok számát előre ismerjük, azok ciklikus beolvasását a FOR-NEXT ciklusutasitáspárral minden gond nélkül elvégezhetjük. Ha például egy N adatot tartalmazó méréssorozat mérési eredményeit akarjuk a gépbe bevinni, akkor a következő programrészt használhatjuk.

```
200 RE! ** MERES ADATAINAK BEVITELE **
210 PRINT "A MERESI ADATOK SZAMA:"
220 INPUT N%
230 ON ERROR GOTO 220
240 DIM A(N%)
250 FOR I%=1% TO N%
260 PRINT "A";I%;"-EDIK MERESI ADAT:"
270 INPUT A(I%)
280 ON ERROR GOTO 270
290 IF A(I%)<100 THEN PRINT "GEPELESI HIBA" ELSE GOTO 310
300 GOTO 270
310 NEXT I%
320 REM ** MERESI ADATOK BEVITELENEK VEGE **
```

Fenti programrészben a 23ø-as utasitással csak a mérési adatok számaként bevitt adat tipusát ellenőriztük. Az intervallumvizsgálat elhagyása miatt a gép negativ mérésszámot is elfogadna. A ciklusban az adatokra vonatkozóan egy tipus és egy félig nyilt intervallum vizsgálatot végzünk. A cikluson belüli vezérlésátadásoknál vigyázni kell arra, hogy egy adat elfogadása után a vezérlés a cikluszáró NEXT utasitásra kerüljön, mert csak igy növekszik a ciklusszámláló értéke.

A mérési sorozat eredményeinek számát előre nem ismerve a beolvasási ciklust csak ugy állithatjuk le, hogy az utolsó érvényes adatot követően olyan értéket viszünk be, amelyről a gép felismeri az adatsorozat végét. Ez a speciális érték a beolvasandó adatok értelmezési tartományában általában megtalálható, de az előfordulásának valószinűsége igen-igen csekély kell, hogy legyen. Mérési adatokat beolvasó példánkban legyen ez a speciális érték (beolvasási ciklus leállitó) 999999. Ha ezt az adatot irjuk be a nagy ismétlési számra beállitott ciklusnak vége szakad, mint az a következő példában látható.

```
350 REM ** MERES ADATAINAK BEVITELE **
370 PRINT "AZ ADATBEVITELT A 999999ERTEK BEIRASAVAL FEJEZHETI BE"
380 FOR I%=1% TO 100%
390 PRINT "A";I%;"-EDIK MERESI ADAT"
400 INPUT A(I%)
410 ON ERROR GOTO 390
420 IF A(I%)=999999 THEN 450
430 IF A(I%)<100 THEN PRINT "GEPELESI HIBA" ELSE GOTO 440
440 GOTO 390
450 NEXT I%
460 N%=I%-1%
470 REM ** MERESI ADATOK BEVITELENEK VEGE **
```

A program automatikusan számolja a bevitt adatokat és a 450-es utasitásban az N% változóhoz hozzá is rendeli a beolvasott értékek számát, amelyet a továbbiakban a FOR ciklusparaméter végértékeként használhatunk.

6.2. Kiirás.

"Mutasd meg az eredménylistát, megmondom ki vagy". Ha még nem is gyökeresedett meg ez a szólás, legalább annyi igazságtartalma van, mint a régieknek - az eredménylistának áttekinthetőnek, világosnak kell lennie-. Ezt a követelményt csak ugy lehet kielégiteni, ha a nyomtatási képet előre megtervezzük, legcélszerübben olyan hálózaton, mint amilyen a képernyő (sor-oszlopszám), ill. a nyomtatott lista (ha van sornyomtatónk).

Ha a terv megvan, jöhet a kivitelezés.

6.2.1. Képernyőtörlés.

A képernyő törlését többféleképpen megoldhatjuk. A legegyszerübb ezt a

```
PRINT CHR$(12)
```

paranccsal vagy utasitással megtenni. Az utasitás végrehajtása után a cursor az 1. sor (Vigyázat, Ø. sor is van!) Ø. poziciójára kerül. Ha a nulladik sorba is akarunk irni, akkor a PRINT CHR$(12) után ;-t kell elhelyeznünk.

6.2.2. Cursor pozicionálás

Cursort pozicionálni (helyre beállitani) a CUR és TAB függvényekkel lehet. Ezeket a függvényeket PRINT utasitásban kell elhelyezni.
A CUR argumentumaként azt a sor és oszlopszámot (vagy azt megadó kifejezést) kell megadni, amelyek a kiirási mező kezdetét meghatározzák.
A TAB a soron belüli cursor pozició megadására használható.

Irjuk meg azt a programrészt, amely a mérési adatokat feldolgozó program kezdete lehetne; a képernyő ø. sorának, 7. oszlopától kezdve irjuk ki a feladat cimét, majd a cim után irjunk ki (!) egy üres sort.

```
40 REM ** FELADAT CIMENEK KIIRASA **
50 PRINT CHR$(12);CUR(0,7);"MERESI ADATOK FELDOLGOZASA"
60 PRINT
70 REM ** CIMKIIRAS VEGE **
```

6.2.3. Hosszu listák kiirása.

A képernyőn viszonylag kevés sor jelenithető meg egyszerre, ezért a sok sort tartalmazó eredménylistákat részletenként kell kiiratni.
Mivel a hosszu listákat általában ciklusban iratjuk ki, gondoskodni kell arról, hogy ez a részletenkénti (mondjuk 19 sort tartalmazó) kiirás történjen meg,
Ha pl. az $y = x \cdot \sin^2 x$ függvény értékeit kell kiiratnunk, miközben x nullától kettőig változik ø,ø5-os lépésközzel, akkor az azt megoldó programrész a következő lehet.

```
100 REM ** FUGGVENY TABELLAZAS **
110 REM ** KEZDOERTEK BEALLITAS **
120 X1=0
130 I=0
140 REM ** FEJLEC IRAS **
150 PRINT CHR$(12);TAB(4);"FUGGVENY TABELLAZAS"
160 PRINT
170 PRINT TAB(2);"X";TAB(17);"Y"
180 PRINT
190 REM ** FV.ERTEK SZAMITAS,KIIRAS **
200 FOR X=X1 TO 2 STEP 0.05
210 Y=X*SIN(X)*SIN(X)
220 PRINT X,Y
230 I=I+1
240 IF I<18 THEN NEXT X
250 REM ** 18 KIIRT SOR UTANI LEALLAS,INDITAS **
260 X1=X+0.05
270 GET V$
280 GOTO 130
290 NEXT X
300 REM ** TABELLAZAS VEGE **
```

6.3. Megjegyzések.

A programozók a programlistát többnyire saját, egy személyre szóló ügynek tekintik és nem is törekszenek arra, hogy a programot bárki megérthesse. Akármilyen kis programról is van szó, mindig célszerü a nagyobb programegységek elején REM utasitásban megadni, hogy mit is old meg a következő programrész. Ezek a megjegyzés sorok nemcsak mások tájékozódását könnyitik meg, hanem bizonyos idő eltelte után magát a programkészitőt is képessé teszik az esetleg régen elfelejtett megoldási eljárás könnyü felelevenitésére.

Függelékek:

F1 Rendszerparancsok és rendszerutasitások

F2 Rendszerfüggvények

F3 Hibajelzések

F1

Az alábbiakban megadjuk az ABC-8Ø BASIC programozási nyelv teljes szintaktikai leirását. A felsorolásban alkalmazott jelölések, röviditések és azok jelentése a következő:

[ ] A szögletes zárójelben szereplő tagok tetszőlegesen elhagyhatók.

{ } A kapcsos zárójelben szereplő tagok közül az egyik, de csak az egyik adandó meg.

| A tagok elválasztására szolgáló jel.

... A három pont előtt álló tag/ok/ tetszőlegesen sokszor szerepeltethető/k/.

arit-kif Aritmetikai kifejezés

kar-kif Karakteres kifejezés

log-kif Logikai kifejezés

arit-vált Aritmetikai változó

kar-vált Karakteres változó

arit-t-vált Aritmetikai tömb változó

kar-t-vált Karakteres tömb változó

CLEAR

Törli az összes változót.

ED sorszám

Szerkesztésre előkésziti a sorszám-mal azonositott sort.

```
                ⎡⎡sorszám                                                    ⎤⎤
                ⎢⎢                                                           ⎥⎥
                ⎢⎢sorszám {,|-} sorszám                                      ⎥⎥
                ⎢⎢                                                           ⎥⎥
LIST            ⎢⎢sorszám {,|-}                                              ⎥⎥
                ⎢⎢                                                           ⎥⎥
                ⎢⎢{,|-} sorszám                                              ⎥⎥
                ⎢⎢                                                           ⎥⎥
                ⎣⎣[{"|'}] eszköznév: [file-név [.file-tipus]][{"|'}]        ⎦⎦
```

A tárban lévő programot a képernyőre, ill. a megadott file-ba listázza.

Megjegyzés: Ha nem adunk meg határokat, a teljes programot kilistázza.

```
LOAD[{"|'}] eszköznév: [file-név [.file-tipus]][{"|'}]
```

Kitörli a tárban lévő programot és a változókat, majd a tárba tölti a megadott program-file-t.

MERGE[{"|'}] eszköznév:[file-név[.file-tipus]][{"|'}]

A tárban lévő programhoz hozzámásolja a meghatározott program-file-t.

NEW

Kitörli a programot és a változókat, valamint lezárja a nyitott file-okat.

REN [{kezdő-sorszám [,lépésköz]}]
[{kezdő-sorszám }]

Átsorszámozza a tárban lévő programot.

Megjegyzés: Ha csak a kezdő-sorszám-ot adjuk meg, azt egyben a lépésköznek is tekinti a rendszer. Alapértelmezés: 1Ø,1Ø

RUN [[{"|'}] eszköznév:[file-név[.file-tipus][{"|'}]]

/Betölti a program-file-t/ törli a változókat, kezdőértéket ad a véletlenszám-generátornak, majd meginditja a tárban lévő program végrehajtását.

RENDSZERPARANCSOK /folytatás/

SAVE [[{"|'}]eszköznév:[file-név[.file-tipus]][{"|'}]]

A megadott eszközre a megadott névvel és file-tipus-sal rögziti a tárban lévő programot.

SCR

Hatása azonos a NEW parancs hatásával.

RENDSZERUTASITÁSOK

CHAIN { kar-kif / kar-vált / kar-t-vált / {"|'}[eszköznév:] file-név[.file-tipus]{"|'} }

Kitörli a tárban lévő programot, a változókat, kezdőértéket ad a véletlenszám-generátornak, majd a tárba tölti a megadott program-file-t és elinditja azt.

CLOSE file-szám

Lezárja a file-szám-mal azonositott file-t.

CLRDOT sorcim,oszlopcim

Kikapcsolja /elsötétiti/ a meghatározott grafikus pontot.

DEFFN arit-vált [(arit-vált | kar-vált)[,arit-vált | kar-vált]...]= arit-kif

Egy felhasználói függvényt definiál.

DIM arit-t-vált (sorindex[,oszlopindex])[=hossz] | kar-t-vált (sorindex[,oszlopindex])[=hossz] [,arit-t-vált (sorindex[,oszlopindex])[=hossz] | kar-t-vált (sorindex[,oszlopindex])[=hossz]]...

Meghatározza a numerikus, ill. karakteres tömb/ök/ méretét.

END

A programvégrehajtást megállitja. A változók értékét törli, a nyitott file-okat lezárja.

FOR ciklus-változó = kezdőérték TO végérték [STEP lépésköz]

A ciklus paramétereit definiálja. /A NEXT utasitással együtt használandó!/

GET { kar-vált / kar-t-vált }

A változóhoz a klaviatúra legközelebb leütésre kerülő billentyüjének kódját rendeli.

GOSUB sorszám

Átadja a vezérlést a sorszám-mal azonositott sorban kezdődő szubrutinnak.

GOTO sorszám

Átadja a vezérlést a sorszám-mal azonositott sorra.

IF log-kif { THEN { sorszám / Basic utasitás [: Basic utasitás]... } / GOTO sorszám } [ELSE { sorszám / Basic utasitás [: Basic utasitás]... }]

Végrehajtja a THEN/GOTO ágat, ha a log-kif aktuális értéke = IGAZ, ellenkező esetben végrehajtja az ELSE ágat, ill. a következő sorban lévő utasitást.

RENDSZERUTASITÁSOK /folytatás/

INPUT [# file-szám,] { arit-vált / arit-t-vált / kar-vált / kar-t-vált } [, { arit-vált / arit-t-vált / kar-vált / kar-t-vált }]...

Az adott változó/k/hoz a beolvasott érték/ek/et rendeli.

INPUTLINE [#file-szám,] { kar-vált / kar-t-vált }

Az adott változóhoz a beolvasott karaktersorozatot rendeli /a CR/LF-et is/.

[LET] { { arit-vált / arit-t-vált } = { log-kif / arit-kif } ; { kar-vált / kar-t-vált } = kar-kif }

Az adott változóhoz az adott kifejezés aktuális értékét rendeli.

NEXT ciklus-változó

A ciklus utolsó utasitása. /A FOR utasitással együtt használandó!/

NOTRACE

Kikapcsolja a nyomkövetést.

ON ERROR GOTO sorszám

Átadja a vezérlést a sorszám-mal azonositott sorra, ha a rendszer egy lekezelhető hibát észlel. /lásd HIBAJELZÉSEK/

ON arit-kif GOTO sorszám [,sorszám]...

Átadja a vezérlést a sorszámlista arit-kif-edik eleme által azonositott sorra.

ON arit-kif GOSUB sorszám [,sorszám]...

Átadja a vezérlést a sorszámlista arit-kif-edik eleme által azonositott sorban kezdődő szubrutinnak.

ON arit-kif RESTORE sorszám [,sorszám]...

Visszaállitja az adatmutatót a sorszámlista arit-kif-edik eleme által azonositott DATA sorra.

OPEN { kar-kif / kar-vált / kar-t-vált / {"|'}[eszköznév:] file-név [.file-tipus]{"|'} } AS FILE file-szám

Megnyitja a file-t olvasásra és a file-hoz a megadott file-szám-ot rendeli.

OUT port-cim,adat-byte

A cimzett port-ra az adat-byte értékét küldi ki.

POKE tár-cim,adat-byte

A tár-cim-re az adat-byte értékét helyezi el.

PREPARE { kar-kif | kar-vált | kar-t-vált | {"|'}[eszköznév:] file-név [.file-tipus] {"|'} } AS FILE file-szám

Megnyitja a file-t irásra és a file-hoz a megadott file-szám-ot rendeli.

{PRINT | ;} [#file-szám {,|;}] [{arit-kif | kar-kif | log-kif | TAB (oszlopcim) | CUR (sorcim,oszlopcim)} [{,|;} {arit-kif | kar-kif | log-kif | TAB (oszlopcim) | CUR (sorcim,oszlopcim)}]] ...

Megjeleniti /vagy a kiválasztott file-ba irja/ a megadott adatokat.

RANDOMIZE

Az RND függvénynek véletlen kezdőértéket ad.

READ { arit-vált / arit-t-vált / kar-vált / kar-t-vált } [, { arit-vált / arit-t-vált / kar-vált / kar-t-vált }]...

Az adott változó/k/hoz a DATA-területről beolvasott érték/ek/et rendeli.

REM [megjegyzés]

Magyarázó megjegyzést helyez a programba.

RESTORE [sorszám]

Visszaállitja az adatmutatót /a sorszám-mal jelzett sorra/.

RETURN

A szubrutin utolsó utasitása, a vezérlést az utoljára végrehajtott GOSUB utasitást követő utasitásnak adja vissza.

SETDOT sorcim,oszlopcim

Bekapcsolja /kivilágositja/ a meghatározott grafikus pontot.

STOP

A programvégrehajtást megállitja. A változók értékei nem törlődnek.

TRACE

Bekapcsolja a nyomkövetést.

RENDSZERFÜGGVÉNYEK

ABS(arit-kif)	Az arit-kif abszolút értéke.
ADD%(kar-kif,kar-kif,arit-kif)	A karaktersorozatok numerikus összege.
ASC(kar-kif)	A karaktersorozat 1. elemének kódja.
ATN(arit-kif)	Az arit-kif arkusz tangensének főértéke radiánban.
CALL(tárcim[,arit-kif])	Gépi szintü szubrutinhivás.
CHR$(arit-kif[,arit.kif]...)	Az arit-kif, mint ASCII kód karakteres megfelelője. Megjegyzés: maximum 4 db arit-kif szerepeltethető.
COMP%(kar-kif1,kar-kif2)	Karaktersorozatok összehasonlitása = -1, ha kar-kif1 < kar-kif2 = Ø, ha kar-kif1 = kar-kif2 = +1, ha kar-kif1 > kar-kif2
COS(arit-kif)	A radiánban értelmezett arit-kif koszinusza.
CUR(sorcim,oszlopcim)	A kurzort a megadott pozicióra állitja.
DIV%(kar-kif,kar-kif,arit-kif)	A karaktersorozatok numerikus hányadosa.
DOT(sorcim,oszlopcim)	A meghatározott grafikus pont állapota.
ERRCODE	Az utolsó hiba kódja.
EXP(arit-kif)	Az arit-kif-el végzett e alapú exponenciális művelet.

RENDSZERFÜGGVÉNYEK /folytatás/

FIX(arit-kif)	Az arit-kif-ben benntfoglalt legnagyobb egész szám.
FN arit-vált [({arit-kif / kar-kif}[,{arit-kif / kar-kif}]...)]	Felhasználói függvényhivás.
INP(port-cim)	Az adott port-ról olvasott adat-byte értéke.
INSTR(arit-kif,kar-kif,kar-kif)	Karaktersorozat előfordulásának kezdőpoziciója.
INT(arit-kif)	Az arit-kif egész része.
LEFT$(kar-kif,arit-kif)	A karaktersorozat balra eső része.
LEN(kar-kif)	A karaktersorozat hossza.
LOG(arit-kif)	Az arit-kif természetes alapú logaritmusa.
LOG10(arit-kif)	Az arit-kif 10-es alapú logaritmusa.
MID$(kar-kif,arit-kif,arit-kif)	A karaktersorozat középső része.
MUL$(kar-kif,kar-kif,arit-kif)	A karaktersorozatok numerikus szorzata.
NUM$(arit-kif)	Az arit-kif karakteres értéke.
PEEK(tárcim)	Az adott tárcim-ről olvasott adat-byte értéke.
PI	π (3.14159)
RIGHT$(kar-kif,arit-kif)	A karaktersorozat jobbra eső része.
RND[(arit-kif)]	A pszeudó véletlenszám-sorozat következő eleme.

RENDSZERFUGGVENYEK /folytatás/

SGN(arit-kif)	Az arit-kif előjele = -1, ha arit-kif < Ø = Ø, ha arit-kif = Ø = +1, ha arit-kif > Ø
SIN(arit-kif)	A radiánban értelmezett arit-kif szinusza.
SPACE$(arit-kif)	Adott hosszúságú, betüközből álló karaktersorozat.
SQR(arit-kif)	Az arit-kif négyzetgyöke.
STRING$(arit-kif,arit-kif)	Adott hosszúságú, meghatározott karakterből álló karaktersorozat.
SUB$(kar-kif,kar-kif,arit-kif)	A karaktersorozatok numerikus különbsége.
SWAP%(arit-kif)	Az arit-kif alsó és felső byte-ját felcseréli.
TAB(oszlopcim)	A kurzort a meghatározott oszlopra állitja.
TAN(arit-kif)	A radiánban értelmezett arit-kif tangense.
VAL(kar-kif)	A karaktersorozat numerikus értéke.

HIBAJELZÉSEK

Ø Meg nem engedett index.

1 Hibás számú index.

2 Parancs módban nem használható.

3 A tároló megtelt.

+ 4 Lebegőpontos túlcsordulás.

5 Tömbhatáron túlmutató index.

6 Nincs ilyen sorszámú sor.

+ 7 Egésszám túlcsordulás.

8 Ebben a rendszerben nem alkalmazható.

9 Karaktersorozaton túlmutató index.

1Ø Túl rövid a karakteres változó maximális hossza.

11 Értelmezhetetlen utasitás vagy parancs.

+12 Egy szám nem számjegy karaktert tartalmaz.

13 Hibás számú vagy hibás tipusú argumentum.

14 Egy utasitás után hibás karakter áll.

15 "=" hiányzik vagy rossz helyen áll.

16 Hiányzik a sorszám.

17 Számok és betük meg nem engedett keveredése.

18 " " hiányzik vagy rossz helyen áll.

+19 A file már nem nyitható meg.

2Ø Túl hosszú sor. /Több, mint 12Ø karakter hosszú./

+21 A file-t nem találja.

22 Meg nem engedett utasitás.

23 Hiányzik a "TO".

24 Hiányzik a "NEXT".

25 Hibás utasitás az "ON" után.

26 Hiba az "ON"-hoz tartozó kifejezésben.

27 "NEXT" "FOR" nélkül fordult elő.

HIBAJELZÉSEK /folytatás/

28 Hibás változó a "NEXT" után.

29 "RETURN" "GOSUB" nélkül fordult elő.

+3Ø Elérte a DATA-listа végét, tovább nem olvasható.

31 A parancs után hibás adatok állnak.

32 A file nincs megnyitva.

33 Hiányzik az "AS FILE" kulcsszó.

+34 Vége a file-nak.

+35 Ellenőrző összeg hiba olvasáskor.

+36 Ellenőrző összeg hiba iráskor.

+37 Hibás formátumban tárolt adatok a diszken vagy szalagon.

+38 A kivánt rekord nem része a file-nak.

+39 A file irásvédett.

+4Ø A file törlésvédett.

+41 Megtelt a diszk.

+42 A diszkegység nem üzemkész.

+43 A diszk irásvédett.

44 A logikai file nincs megnyitva.

45 Hibás logikai file szám.

46 Hibás diszkegység szám.

47 Hibás csapda sorszám.

48 Könyvtárhiba a diszken.

49 Hibás fizikai file szám.

5Ø Gyökvonás negativ számból.

51 Készülék hiba.

52 Érvénytelen utasitás az adott eszközre.

53 Hibás sor.

54 IEC adó/vevő hiba.

55 IEC vevő nem aktiv.

HIBAJELZÉSEK /folytatás/

56 IEC adó nem aktiv.

57 Nem definiált felhasználói függvény.

58 Érvénytelen karakter.

59 Incompatibilis program formátum.

6Ø Tárcim 65535.

61 Hiányzik egy vessző.

62 sorcim, ill. oszlopcim a képernyőn kivülre mutat.

63 Hiányzik az "AS" kulcsszó.

64 Hibás "RENAME" utasitás.

65 Karakteres /ASCII/ aritmetikai túlcsordulás.

66 A karaktersorozat nem csupán számjegykaraktereket tartalmaz egy ASCII aritmetikai müveletben.

A "+" jellel megjelölt hibajelzések egy ON ERROR GOTO utasitással lekezelhetők.